PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-092796

(43)Date of publication of application : 06.04.2001

(51)Int.CI. G06F 15/167

G06F 12/02

G06F 12/06

G06F 12/08

G06F 12/10

G06F 13/16

G06F 15/16

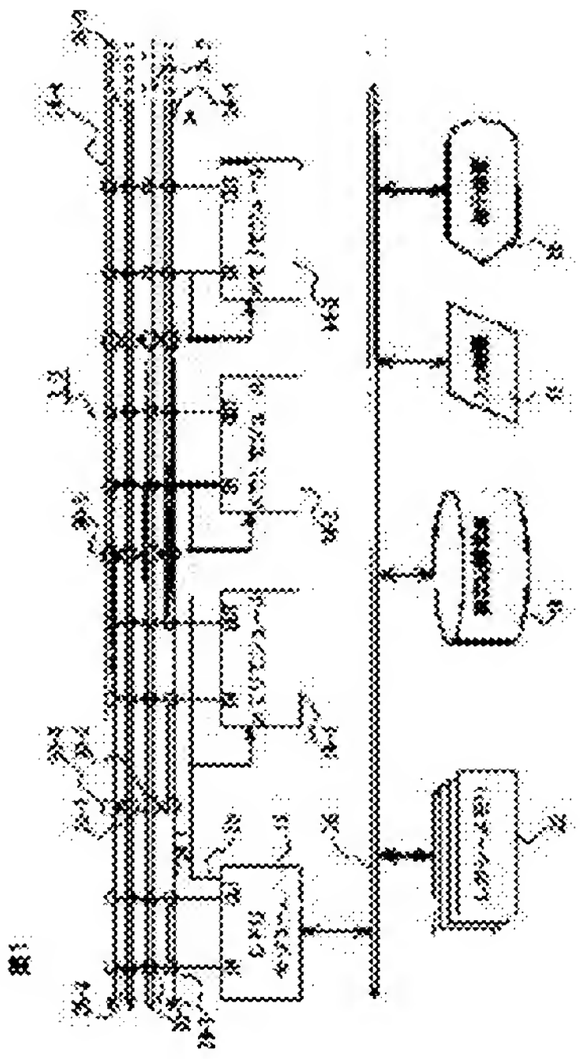
(21)Application number : 11-263793

(22)Date of filing : 17.09.1999

(71)Applicant : TAABO DATA LABORATORY KK

(72)Inventor : KOSHO SHINJI

(54) ARCHITECTURE OF PARALLEL COMPUTERS AND INFORMATION PROCESSING UNIT USING THE SAME



(57)Abstract:

PROBLEM TO BE SOLVED: To provide a computer architecture capable of realizing an extremely high speed parallel processing in a distributed memory type.

SOLUTION: A computer system 10 is provided with a CPU module 12, plural memory modules 14 each of which has an MPU 36 and a RAM core 34, plural sets of buses 24 to connect a CPU with the memory module and to connect between memory modules and each memory module is operated by an instruction to be given from the CPU 12. A space ID is imparted to a series of data with specified relation, each memory module manages a table at least including the space ID, a logical address regarding a part of a series of data to be managed by itself, the size of a series of data, judges whether the part of a series of data to be managed by itself concerns in the accepted instruction or not and executes a processing regarding the data stored in the RAM core.

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against

examiner's decision of rejection]

[Date of extinction of right]

CLAIMS

[Claim(s)]

[Claim 1] The connection between a CPU module, two or more memory modules with which each has MPU and a RAM core, and said CPU and memory module, and/ By or the instruction which is equipped with two or more sets of buses which connect between memory modules, and is given to MPU of each memory module from CPU It is the architecture of the parallel computer constituted so that MPU of each memory module might operate. Space ID is given to a series of data which have predetermined relation. MPU of each memory module The logical address about the part of a series of data which the space ID concerned and self manage at least, It judges whether the part of a series of data which self manages is participating in the instruction which managed the table containing the size of the part concerned, and the size of a series of data, and MPU of each memory module received. the architecture of the parallel computer characterized by being constituted so that reading appearance of the data memorized by the RAM core may be carried out, may send out to a bus, the data given through the bus may be written in a RAM core, and processing required for data may be performed and/or said table may be updated.

[Claim 2] The computer architecture according to claim 1 characterized by having the address comparator with which said MPU compares the space comparator in comparison with the space ID of a series of one or more data with which self manages the space ID given from CPU, and the logical address given from CPU with the logical

address of the part of the data which self manages, and address KARIKYURETA which computes the physical address on a self RAM cell based on the logical address concerned.

[Claim 3] The input in which connection with or [any of two or more sets of said buses] is [that each of said memory module accepts the synchronizing signal for synchronizing a CPU module and other memory modules] possible, It has the output in which connection with or [other any of said bus which are groups] is possible, and said synchronizing signal is followed at least. By connection with said which bus and input The computer architecture according to claim 1 or 2 characterized by being constituted by connection between which bus besides the above, and an output, inputting data so that data can be outputted.

[Claim 4] To each of said bus which are groups, between the input of said CPU module and which memory module, or outputs, and/ The switch for specifying connection between the input of which other memory modules or an output, and the output of the memory module of further others or an input is formed. Or by change-over of said switch The computer architecture according to claim 3 characterized by realizing transfer of data in juxtaposition in each of two or more sets of buses.

[Claim 5] Into said 1st bus which it is in any of the buses of a group, the output of which memory module, the input of which other memory modules connects -- having -- and said 2nd bus which it is in other any among the buses of a group -- being concerned -- others -- with the output of which memory module The computer architecture according to claim 4 to which the input of which memory module of further others is connected, and transfer of the data in the 1st bus and transfer of the data in the 2nd bus are characterized by going on in juxtaposition.

[Claim 6] The computer architecture according to claim 5 characterized by repeating connection between said buses and memory modules, and forming connection between multistage memory modules.

[Claim 7] Said MPU deletes the specific element in a series of data, and a specific element is inserted into said a series of data. If the instruction which shows that a specific element is added to the tail of a series of data is received A computer architecture given in claim 1 characterized by comparing with the location of the element concerning deletion, insertion, or an addition the field of the data which self manages with reference to a table, and updating the content of said table according to the comparison result concerned thru/or any 1 term of 6.

[Claim 8] A computer architecture given in claim 1 characterized by performing value conversion said whose MPU answers the given instruction, and changes the suffix for

specifying the element in a series of data, and/or gives specific qualification to an element thru/or any 1 term of 7.
[Claim 9] The connection between a CPU module, two or more memory modules with which each has MPU and a RAM core, and said CPU and memory module, and/ By or the instruction which is equipped with two or more sets of buses which connect between memory modules, and is given to MPU of each memory module from CPU It is the information processing unit constituted so that MPU of each memory module might operate. Space ID is given to a series of data which have predetermined relation. MPU of each memory module The logical address about the part of a series of data which the space ID concerned and self manage at least, It judges whether the part of a series of data which self manages is participating in the instruction which managed the table containing the size of the part concerned, and the size of a series of data, and MPU of each memory module received. the information processing unit characterized by being constituted so that reading appearance of the data memorized by the RAM core may be carried out, may send out to a bus, the data given through the bus may be written in a RAM core, and processing required for data may be performed and/or said table may be updated.
[Claim 10] The information processing unit according to claim 9 characterized by constituting said CPU module possible [the bus of the others which interconnect legacy memory, an input device, and an indicating equipment, and connection].
[Claim 11] The computer system characterized by having the storage, input unit, and display containing one or more legacy memory connected through an information processing unit according to claim 9, and a CPU module and other buses.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Industrial Application] This invention relates to the computer architecture in which general-purpose parallel operation is possible by memory control more suitable for a detail, and high-speed about the architecture of the parallel computer which can realize SIMD (Single Instruction Stream, Multiple Data Stream).

[0002]
[Description of the Prior Art] A computer is introduced into various locations of the whole society, and large-scale data came to be stored here [there] by the end of today when networks including the Internet permeated. In order to process such large-scale data, huge count is required, therefore trying to introduce parallel processing is natural.
[0003] Now, parallel processing architecture is divided roughly into a "shared memory mold" and a "distributed memory type." The former ("shared memory mold") is a method with which two or more processors share one huge room. Since the traffic between a processor group and a shared memory serves as a bottleneck by this method, it is not easy to build a realistic system using the processor exceeding 100. In case the square root of 1 billion floating point variables is calculated by following, the acceleration ratio to a single CPU will call it at most 100 times. Experientially, about 30 times is an upper limit. Each processor has respectively local memory, and the latter ("distributed memory type") combines these, and builds a system. The design of the hardware system which also incorporated hundreds - tens of thousands of processors by this method is possible. Therefore, it is possible to make the acceleration ratio to the single CPU at the time of calculating the square root of the 1 billion above-mentioned floating point variables one 10,000 times the number [hundreds -] of this. However, some technical problems mentioned later exist also in the latter. This application is made to perform the comparison with the conventional technique about a "distributed memory type", adding some considerations first about this method.
[0004]
[Means for Solving the Problem] [-- 1st technical-problem: -- the 1st technical problem of division-of-responsibilities management] "a distributed memory type" of a huge array is a problem of division-of-responsibilities management of data. Huge data (since it is generally an array, an array explains henceforth) cannot be held in the local memory which one processor owns, and division-of-responsibilities management is inevitably carried out at two or more local memories. If an efficient and flexible division-of-responsibilities management mechanism is not introduced, it is clear that various failures will be held on the occasion of development and activation of a program.
[0005] [-- 2nd technical-problem: -- the effectiveness of interprocessor communication -- low -- as for access to the array element which other processors own, interprocessor communication is made indispensable although it can access

promptly to the array element on the local memory which self owns, if each processor of] distribution memory mold system tends to access a huge array. It is said that this interprocessor communication has extremely low performance, and is cut in 100 clocks also at the lowest compared with the communication link with a local memory. For this reason, at the time of sort implementation, since reference covering the huge array whole region is carried out and interprocessor communication occurs frequently, performance falls extremely.

[0006] Explanation is more concretely added about this trouble. 1999 current and a personal computer are constituted as a "shared memory mold" using CPU of 1 - some. It operates with an about 5 to 6 times [of a memory bus] internal clock, that interior is equipped with the automatic parallel execution function or the pipeline processing function, and standard CPU used for this personal computer can process about 1 data with one clock (memory bus). When the personal computer which is a "shared memory mold" performs sorting application of a huge array, demonstrating the multiprocessor system 100 times the performance of a "distributed memory type" which requires one clock about one data and one data takes 100 clocks (memory bus) for this reason is also considered.

[0007] The 3rd technical problem of [supply "a distributed memory type" of the technical-problem:program which is the 3rd] is a problem how to supply a program to many processors. Dramatically, a separate program is loaded to many processors and they take a great quantity of loads for creation of a program, compile, and distribution by the method (MIMD:Multiple Instruction Stream, Multiple Data Stream) which carries out coordination actuation of the whole. On the other hand, by the method (SIMD:Single Instruction Stream, Multiple Data Stream) which operates many processors by the same program, the degree of freedom of a program decreases and the situation where the program which brings about the result of a request cannot be developed is also assumed.

[0008] This invention offers the approach and computer architecture which solve the above 1st of a "distributed memory type" thru/or the technical problem of 3. The technical problem of the 1st "division-of-responsibilities management of a huge array" can solve arrangement (physical address) of each element of an array by carrying out division-of-responsibilities management by the approach with each unific processor module. By this technique, the need for a garbage collection can be lost and the tacit processing (un-explicit) assignment of each processor indispensable when insertion and deletion of an array element are completed with a number clock and realizes SIMD can also be assigned. This approach is explained like after by the

concept of "multi-space memory."
[0009] the 2nd effectiveness of "interprocessor communication -- low -- " -- a technical problem can switch each interprocessor according to the processing which it be going to attain, it be the sequence that the data of the defined class be able to be defined, for every connection path, it schedule-ize a communication link so that the capacity of a bus may be use in the one direction to about 100% by carry out a continuation transfer, and it can solve it by realize huge pipeline processing simultaneously. The back will illustrate the structure-of-a-system approach which completes the sort of 1 billion lines in about 1 second by the realistic system design, in order to prove the effectiveness. This is high-speed 10,000 or more times compared with the equipment of the known maximum high speed. This approach is explained as a "recombination bus" technique like after.
[0010] The technical problem of the 3rd "supply of a program" is solvable by adopting a SIMD method. Although the tacit processing (un-explicit) assignment of each processor is determined how in the case of SIMD or ? is the biggest problem, this processing assignment can be automatically determined by the above-mentioned "multi-space memory" technique, and even if it is SIMD, the degree of freedom of a program can be held. That is, this invention outputs and inputs the element under array memorized by various memory with a single instruction, and aims at offering the computer architecture which can realize high-speed remarkable parallel processing in a distributed memory type.
[0011]
[Means for Solving the Problem] The connection between two or more memory modules with which, as for the object of this invention, a CPU module and each have MPU and a RAM core, and said CPU and memory module, and/ By or the instruction which is equipped with two or more sets of buses which connect between memory modules, and is given to MPU of each memory module from CPU It is the architecture of the parallel computer constituted so that MPU of each memory module might operate. Space ID is given to a series of data which have predetermined relation. MPU of each memory module The logical address about the part of a series of data which the space ID concerned and self manage at least, It judges whether the part of a series of data which self manages is participating in the instruction which managed the table containing the size of the part concerned, and the size of a series of data, and MPU of each memory module received. The data memorized by the RAM core are read, it sends out to a bus, the data given through the bus are written in a RAM core, and processing required for data is performed. And/ Or it is attained by the

architecture of the parallel computer characterized by being constituted so that said table may be updated.

[0012] According to this invention, in order to grasp a series of data using Space ID, even if a series of data concerned are alloted by many memory modules, MPU of each memory module can recognize a series of data concerned certainly. Moreover, since the memory module grasps on the table the part which a series of data and self manage, it can perform predetermined processing with reference to the table according to acceptance of an instruction. Thereby, the parallel processing in each MPU based on a single instruction is realizable.

[0013] In the desirable embodiment of this invention, MPU has the address comparator which compares the space comparator in comparison with the space ID of a series of one or more data with which self manages the space ID given from CPU, and the logical address given from CPU with the logical address of the part of the data which self manages, and address KARIKYURETA which computes the physical address on a self RAM cell based on the logical address concerned. These comparators and KARIKYURETA may consist of hardware, and may be realized by the program of MPU as software.

[0014] Moreover, it sets in the desirable embodiment of this invention. The input in which connection with or [any of two or more sets of said buses] is [that each of a memory module accepts the synchronizing signal for synchronizing a CPU module and other memory modules] possible, said -- while it has two or more outputs in which connection with or [other any of the bus of a group] is possible and data are inputted by connection with said which bus and input at least according to said synchronizing signal -- said -- others -- it is constituted by connection between which bus and an output so that data can be outputted. According to the gestalt of this operation, according to a synchronizing signal, the data output from a memory module and the data input to a memory module are made, and control of connection of a bus enables it to realize parallel processing appropriately.

[0015] It is more desirable that the switch for specifying connection between the input of said CPU module and which memory module or an output and/or between the input of which [other] memory module or an output, and the output of the memory module of further others or an input is formed in each of two or more sets of buses, and transfer of data is realized by change-over of a switch in juxtaposition in each of two or more sets of buses. This becomes possible to use two or more sets of buses for validity more, and it becomes possible to raise parallelism more.

[0016] In the still more desirable embodiment of this invention Into the 1st bus which

it is in any of two or more sets of buses, the output of which memory module, the input of which other memory modules connects -- having -- and said 2nd bus which it is in other any among the buses of a group -- being concerned -- others -- with the output of which memory module The input of which memory module of further others is connected, and transfer of the data in the 1st bus and transfer of the data in the 2nd bus advance in juxtaposition. Thus, according to the embodiment of a computer, a CPU module and a memory module enable it to realize pipeline processing. It is more desirable to repeat connection between a bus and a memory module and to form connection between multistage memory modules.

[0017] It sets in the another desirable embodiment of this invention, MPU deletes the specific element in a series of data, and a specific element is inserted into said a series of data. If the instruction which shows that a specific element is added to the tail of a series of data is received, the field of the data which self manages will be compared with the location of the element concerning deletion, insertion, or an addition with reference to a table, and the content of said table will be updated according to the comparison result concerned. That is, in MPU, it becomes possible to realize deletion, insertion, and an addition of an element by updating the table which self manages, namely, carrying out RIMAPPINGU.

[0018] In still more nearly another embodiment of this invention, value conversion whose MPU answers the given instruction, and changes the suffix for specifying the element in a series of data, and/or gives specific qualification to an element is performed. Moreover, two or more memory modules with which, as for the object of this invention, a CPU module and each have MPU and a RAM core, The connection with said CPU and memory module, and/ By or the instruction which is equipped with two or more sets of buses which connect between memory modules, and is given to MPU of each memory module from CPU It is the information processing unit constituted so that MPU of each memory module might operate. Space ID is given to a series of data which have predetermined relation. MPU of each memory module The logical address about the part of a series of data which the space ID concerned and self manage at least, It judges whether the part of a series of data which self manages is participating in the instruction which managed the table containing the size of the part concerned, and the size of a series of data, and MPU of each memory module received. it is attained by the information processing unit characterized by being constituted so that reading appearance of the data memorized by the RAM core may be carried out, may send out to a bus, the data given through the bus may be written in a RAM core, and processing required for data may be performed and/or said table

may be updated. For example, said unit may be formed in the single circuit board, and the CPU module may be constituted possible [the bus of the others which interconnect legacy memory, an input device and an indicating equipment, and connection].

[0019] Furthermore, the object of this invention is attained by the computer system characterized by having the storage, input unit, and display containing one or more legacy memory connected through the above-mentioned information processing unit, and a CPU module and other buses.

[0020]

[Embodiment of the Invention] With reference to an accompanying drawing, explanation is added per gestalt of operation of this invention below a [hardware configuration]. Drawing 1 is a block diagram which shows the configuration of the computer system concerning the gestalt of operation of this invention. As shown in drawing 1 R> 1, a computer system 10 The CPU module 12 which realizes parallel operation by single instruction, the memory module 14-1 which memorizes various data required for parallel operation, 14-2, 14-3, and --, It has the fixed memory 16 which memorizes a required program and data, the input devices 18, such as a keyboard and a mouse, the indicating equipment 20 which consists of CRT etc., and the legacy memory 22 the data of various formats etc. are remembered to be. Moreover, in a bus 24-1, 24-2, and --, transfer of the information between the circuit elements which a switch 28-1, 28-2, 28-3, --, etc. were arranged by the contact with the CPU module 12 and each memory module 14, and were chosen as it is possible. Moreover, the switch 30-1 for making connection and connection of a bus, 30-2, and -- are prepared between adjoining memory modules between the CPU module 12 and the memory module 14-1.

[0021] Between the CPU module 12 and the memory module 14, two or more buses 24-1, 24-2, 24-3, 24-4, and -- are prepared. Therefore, transfer of data etc. is possible for between the CPU module 12 and a memory module 14 and between memory modules by the above-mentioned bus. Moreover, the control signal line 25 is formed between CPU12 and a memory module 14, and the instruction emitted from CPU12 is transmitted to all the memory modules 14.

[0022] Furthermore, the local bus 26 is arranged between CPU12 and other components, and transfer of data etc. is possible for for example, fixed memory 16, an input device 18, etc. also among these. CPU12 reads the program memorized by other storage (not shown) like RAM which was memorized by fixed memory 16 or was connected on the bus 26, and performs control of the switches 28 and 30 besides

transfer of data including sending out of the instruction to the memory module 14 shown below etc. according to this program. Moreover, CPU12 can receive the data of the various formats memorized by the legacy memory 22 according to a program, can change them into a series of data (array) which can process the data of this format by the system which consists of CPU12, a memory module 14, and a bus 24, and can also store these in each memory module 14.

[0023] Drawing 2 is a block diagram which shows the outline of each memory module 14. As shown in drawing 2 , a memory module 14 The clock buffer 32 which accepts synchronizing signals, such as a clock given from the CPU module 12, The space ID mentioned later, the element number of data, etc. are grasped as the RAM core 34 which memorizes data. MPU36 which controls data read-out from the data writing and RAM core to the RAM core 34 based on Space ID and an element number when the instruction from CPU12 etc. is received, It has I/O38 which receives the data from either of the buses, and supplies the RAM core 34, and/or sends out the data from the RAM core 34 to which bus. In the gestalt of this operation, through the control signal line 25, a memory module 14 accepts the instruction from CPU, MPU36 can answer this instruction, and can read the data of the RAM core 34, and it can write data in the RAM core 34, or can perform predetermined processing now to data. Moreover, a data input and data output are performed based on synchronizing signals, such as a clock given to the clock buffer 32, through the data access to the RAM core 34, and I/O.

[0024] In this invention, it is possible that a computer system 10 is a system of a memory share mold so that clearly from drawing 1 and drawing 2 . Moreover, each memory module 14 performs processing in juxtaposition by giving an instruction to each memory module 14 through the control signal line 25 so that it may mention later. Moreover, the data output to a bus, the data input from a bus, etc. are performed based on a predetermined synchronizing signal. Therefore, it is possible that this computer system 10 is making the gestalt of SIMD.

[0025] [the outline of a function realized] -- before adding explanation more detailed than per [which has such a configuration] computer system 10, the outline of a function realized according to this computer system 10 is explained briefly.

(1) In a multi-space memory book description, multi-space memory means the room assigned in order to access room based on Space ID and the address. Thereby, even if a series of data are alloted by many processors, each processor can separate and recognize this certainly. In the conventional room, even if it might assign the field according to individual for every process, assigning graduation space to every [a series of] variables (an array, structure, etc.) was not performed. Therefore, such

conventional room is hereafter called "single room." In the system of single room, since data are accessed only using the address, a series of data which have relation were not able to be separated, and it has not recognized. For this reason, even if parallel processing was actually possible, that propriety was not able to be judged in many cases. Moreover, the garbage collection needed to be performed in order to secure the hold location of a series of data concerned, when making a series of new data hold in a certain single room.

[0026] On the other hand, in this invention, Space ID was introduced into room and the same ID is given to it about a series of data. Moreover, in a memory module 14, the space ID about the data currently held at the own RAM core 34 can be grasped, and, thereby, each [memory module 14 / itself] can determine the right or wrong of self actuation by referring to the space ID of the data accessed now. Moreover, since each memory module relates with Space ID and all or some of a series of data can be held, a certain data of a series of can be divided and stored in two or more memory modules 14, and, thereby, a garbage collection can be made unnecessary.

[0027] For example, as shown in drawing 3 , in single room, the case where a series of data "A", a series of data "B", and -- are held is considered. For example, total memory size assumes that total of the size of the data of a up Norikazu ream is 30 words by 32 words here. Since it is dotted with the data of these single strings all over space, although intact memory size is 12 words, the size of a series of actually storable data is limited to 3 words. For this reason, a garbage collection must be performed when a series of new data which have the size exceeding 3 words should be held. As shown in one of these, and drawing 4 , Space ID is given to each of a series of data in this invention. These are related with Space ID and memorized by one or more memory modules 14. Therefore, it becomes possible to make in agreement intact size and the size which can be held.

[0028] (2) In a memory module and this invention, each memory module 14 had MPU36, and grasp each element number of a series of data which self besides the above-mentioned space ID holds. Therefore, after receiving the instruction from CPU12, the data which MPU36 should access according to an instruction can judge whether it is what is held in the RAM core 34 of self, and can determine the right or wrong of the need as access. Furthermore, each memory module 14 is able to determine the assignment range of the tacit processing in the instruction in SIMD from the range of the suffix of the array element stored in the RAM core 34 of self.

[0029] Moreover, in this invention, a memory module 14 can perform ADORESURI mapping now. For example, the element of a position is deleted when inserting a

specific element in the position of a certain array, as shown in drawing 5 . Also when adding a predetermined element to the tail of an array, in the gestalt of this operation, MPU36 can realize these at a juxtaposition-and high speed by performing ADORESURI mapping in each holding the element relevant to the array concerned of a memory module. Furthermore, as shown in drawing 6 , also when giving qualification to an array element (value), in each holding the array element related (for example, when adding "1" to each value) of a memory module, MPU36 can perform juxtaposition processing required for a high speed.

[0030] Moreover, in a memory module 14, MPU36 can grasp each size of the data which should be memorized with the RAM core 34, and can memorize these with the compressed gestalt. For example, when a data value actual when the data of an integral value should be held with a certain memory module 14 cannot take only the value to "0" thru/or "3", MPU36 prepares only 2 bits for each data. In order to express one integer, when 32 bits is being used, for the communication link between a memory module 14 and CPU12, MPU36 changes data format, and CPU12 should be delivered [just] between CPUs12 and received. This becomes possible to use the RAM core 34 without futility. Moreover, also about the data with which die length like a character string differs, a data length can be changed similarly and it can memorize now.

[0031] Furthermore, in a memory module 14, a specific value (for example, "0") can be set now to the data related with the predetermined space ID, and the data to which the element number of the predetermined range was given. This becomes possible [performing processing of initialization at a high speed] within a memory module 14. Moreover, in a memory module 14, it is possible to search the value in a certain specific data (array) or to check the range of a subscript.

[0032] (3) Pipeline processing is realized by setting to rearrangeable bus this invention, and CPU's12 turning on / turning off selectively a switch 28-1, 28-2, -- and a switch 30-1, 30-2, and --, and specifying the memory module 14 which should deliver and receive data. As shown in drawing 7 , for example, the data outputted from certain memory module 14-i other memory module 14-j -- giving -- and -- being concerned -- others, when the data outputted from memory module 14-j should be transmitted to memory module 14-k of further others CPU12 sets up the condition of each switch so that bus 24-m may be assigned for memory module 14-i and 14-j and bus 24-n may be assigned for memory module 14-j and 14-k.

[0033] Furthermore, not only when connection between single memory modules realizes, but these pipeline processing can be realized by connection between two or

more of a series of memory modules (memory module group). According to the processing which it is going to attain, between each memory module can be switched, and a communication link can be schedule-ized so that the capacity of a bus can be used for an one direction about 100% by carrying out a continuation transfer in the sequence that the data of the defined class were able to be defined, for every connection path. Thereby, the lowness of the performance of interprocessor communication which was the biggest problem of the parallel processing system of a distributed memory type is cancelable. Thus, in the constituted computer system 10, explanation is added per actuation of the system in the concrete configuration of multi-space memory, and multi-space memory.

[0034] [Multi-space memory] drawing 8 is drawing for explaining the structure of a memory module 14 under multi-space memory. As shown in drawing 8 (a), a space ID managed table is prepared in the RAM core 34 in a memory module 14. Thereby, MPU36 of a memory module 14 becomes possible [grasping required information, such as the space ID of the data which self holds,]. As shown in drawing 8 (b), the logic starting address of a data constellation under the management of Space ID and CPU for every data constellation which self holds, the size of the field where the data constellation was assigned, the physical starting address in the RAM core 34, the total size of a series of data which have the space ID concerned, and the access-restriction flag that shows access restriction are stored in the space ID managed table. in the gestalt of this operation, reading appearance of the access-restriction flag is carried out, and a chisel is possible for it -- only (R) and writing are possible -- (R) and R/W are possible -- three conditions of (RW) can be shown now.

[0035] When the data constellation which has a certain space ID is given, MPU36 of a memory module 14 finds out one or more fields which should hold the data constellation concerned into the RAM core 34, divides a data constellation into remaining as it is or 2 or more, and holds it in the field concerned. In this case, the logic starting address and allotment area size in the RAM core which held data actually with the given space ID, a logic starting address, total size, and an access-restriction flag are also memorized by the space ID managed table. Drawing 8 (c) is drawing showing the data in the RAM core 36 according to the space ID managed table by drawing 8 (b).

[0036] Explanation is added to below per [to [the approximate account of memory access], thus the constituted memory module 14] access. As shown in drawing 9 , CPU12 transmits a required instruction (for example, writing and read-out of data) to

all the memory modules 14 through the control signal line 25 first at Space ID and the logical address, and a list. In each memory module 14, this is answered, the space comparator 52 formed in MPU36 compares Space ID with the space ID currently held on the space ID managed table of self, and it judges whether self holds the same thing, and a address comparator 54 makes the same judgment about the logical address. Subsequently, when it is judged that the data with which MPU36 of a memory module 14 serves as a processing object by the instruction at the RAM core 34 of self are held, with reference to a space ID managed table, address KARIKYURETA 56 computes the physical address in the RAM core 34, and specifies the data used as a processing object. Thus, after data are specified, MPU36 performs processing (for example, writing and read-out of data) according to the instruction given from CPU12, and when required, it transmits data to CPU12 (refer to drawing 9 (c)).

[0037] [-- more concrete : of multi-space memory of operation -- it explains below per [to the condition that the specific element was deleted from the condition that] (this is hereafter called an "array" by the case.), for example, a series of data with a certain space ID, such as deletion of the element under array, was held in one or more memory modules 14] a series of actuation. In certain memory module 14-i, the data constellation belonging to Space ID "010" is stored as shown in drawing 10 (a), and in memory module 14-j of **, the data constellation belonging to Space ID "010" considers the case where it is stored as shown in drawing 10 (b). For example, in memory module 14-i, it turns out that the data from the logical address "0" to "59" are memorized from the physical address "100" of the RAM core. It seems that in this case, an apparent array is shown in drawing 10 (c).

[0038] Thus, when a certain array is stored in two or more memory modules, it states below per processing at the time of deleting a specific element. The case where the instruction of deleting the element "50-59" of Space ID "010" through the control signal line 25 to each memory module 14-1, 14-2, and -- is emitted from CPU12 is considered. Drawing 11 and drawing 13 are flow charts which show the processing performed with each memory module which received the instruction of deleting the element of the predetermined range in a certain space ID.

[0039] MPU36 of each memory module receives the instruction given through the control signal line 25, interprets the content (step 1101), investigates the "space ID" in an instruction (step 1102), and judges whether it relates to the space ID of the data which the RAM core 34 of self holds (step 1103). When judged as a no (No) at step 1103, the size of the range which ends processing, the writing of the data constellation about the space ID concerned of MPU36 has become possible with

reference to the space ID managed table on the other hand when [that] judged yes (Yes), or had the deletion demand judges whether it is smaller than total size (step 1104). When it is judged by the check that it is abnormal (it is yes (Yes) at step 1105), MPU36 notifies that the error arose through the control signal line 25. On the other hand, in [that] being normal, MPU36 compares the range of which deletion was required by the instruction with the range of the element held with the RAM core 34 of self (step 1107), and performs (step 1108) and various processings by the comparison result.

[0040] First, rather than the self range of the element to hold, when the range with a deletion demand is back (refer to "A" and drawing 12 (a) of drawing 11), as for MPU36, it does not perform processing at all (step 1109 reference). When the range with a deletion demand is lapped and located behind the element which self holds (refer to "B" and drawing 12 (b) of drawing 11), MPU36 updates allotment area size (step 1110). That is, allotment area size is changed so that from the head (arrow-head 1201 reference) of the deletion demand range to the tail (arrow-head 1202 reference) of the range of the element held with the RAM core 34 of self may serve as garbage.

[0041] On the other hand, rather than the self range of the element to hold, the range with a deletion demand updates a logic starting address so that it may reduce by the size in which the deletion demand of a logic starting address had MPU36, in being the front (refer to "C" and drawing 12 (c) of drawing 11) (step 1111). furthermore, rather than the range of the self element to hold, when it is the front and only a part laps (refer to "D" and drawing 12 (d) of drawing 11), the range with a deletion demand MPU36 changes a physical starting address into the physical address corresponding to the value "+1" of the tail of the range with a deletion demand while changing a logic starting address into the value of the head of the range with a deletion demand (step 1112). Subsequently, MPU36 updates allotment area size (step 1113).

[0042] Moreover, when the range with a deletion demand includes the range of the element which self holds (refer to "E" and drawing 12 (e) of drawing 11), MPU36 deletes the various data about the space ID concerned from a space ID managed table (step 1114 of drawing 13). When the range which had the deletion demand in the last is included by the range of the element which self holds (refer to "F" and drawing 12 (f) of drawing 11), a space ID managed table is divided into two, and the thing about the various data about back is generated from various data concerning the front from the deletion range, and the deletion range (step 1115). Or MPU36 may carry out time of day of the garbage collection about self RAM34.

[0043] Thus, a single instruction (deletion instruction of a certain space ID) is

answered from CPU12, each memory module 14 operates, and processing required of a predetermined memory module is performed in juxtaposition. Next, when adding a certain element to the tail of an array which has a certain space ID, it explains briefly per. Drawing 14 is a flow chart which shows the processing performed with each memory module which received the instruction of adding an element to the tail of the array of a certain space ID. Step 1401 of drawing 14 – step 1406 are equivalent to step 1101 of drawing 11 – step 1106. Subsequently, it judges whether MPU36 of each memory module 14 should memorize the element which should be added to the RAM core 34 of self (step 1407). MPU36 can realize this by referring to the space ID managed table of self. When judged yes (Yes) at step 1407, the required value in a space ID managed table is updated (for example, allotment area size is changed according to the number of elements to add), and, subsequently the element which should be added to the predetermined field in a RAM cell is written in (step 1409). Or the various values of a space ID managed table are generated, and the element which should be added may be written in the field in a corresponding RAM cell.

[0044] Subsequently, MPU36 updates the value of "the total size" relevant to the space ID concerned in a space ID managed table (step 1410). Also when judged as a no (No) in step 1407, the value of "the total size" to which it relates in a space ID managed table is updated. Also when adding an element to the location of the arbitration under array, processing of a deletion demand and an abbreviation EQC is performed with each memory module 14.

[0045] When combining two or more arrays, or dividing a single array into two or more arrays as shown in drawing 15 (b) as shown in [more concrete association of multi-space memory of : array of operation and division], next drawing 15 (a), explanation is added per. In the computer system 10 concerning the gestalt of this operation, the array which has a certain space ID (it sets to drawing 15 (a) and is Space ID "100"), and/or the array which has other space ID (it sets to drawing 15 (b) and is Space ID "100") may be held in the RAM core of a single memory module, or may be held in the RAM core of two or more memory modules. Drawing 16 is drawing showing the condition that these were held into the memory module by the array and list which have the array which has Space ID "10", and Space ID "11." In drawing 16 (a), the array 1501 whose space ID of the is "10" and whose size of each element is 10 words is shown. The element under this array 1501 is held in a memory module 14-1 thru/or 14-x. Moreover, in drawing 16 (b), the array 1510 whose space ID of the is "11" and whose size of each element is 10 words is shown. The element of this array 1510 is also held in a memory module 14-1 thru/or 14-x.

[0046] If CPU12 minds the control signal line 25 and emits the instruction of a purport [Space / of "space ID "10" / the array and Space ID "the array of 11" is combined"], each memory module 14 will receive this and will judge whether it is an instruction about the space ID of the data holding self. These processings are the same as that of step 1101 of drawing 11 thru/or step 1106, and abbreviation. Subsequently, when the space ID of the data holding self relates to the instruction, MPU of a memory module realizes association of an array according to the following procedures. Each memory module 14 related when shown in above-mentioned drawing 16 updates the value of the space ID managed table about Space ID "11", when the element of the both sides of Space ID "10" and Space ID "11" is held. More specifically with reference to the value of "the total size" about Space ID "10", the logic starting address is computed again (for example, the sign 1701 of drawing 17 , 1702 reference). Moreover, each related memory module updates the value of "the total size" in a space ID managed table to the thing corresponding to the size which considered two arrays (for example, sign 1703 reference of drawing 17). Drawing 17 is the array 1710 acquired by doing in this way, and drawing showing the space ID managed table (for example, a sign 1711, 1712 reference) in each memory module 14-1 to 14-x.

[0047] Drawing 18 is drawing showing an example divided into the array which has Space ID "10" for the array which has Space ID "10", and the array which has Space ID "11." The decomposition point of the array which has Space ID "10" shown in drawing 18 R> 8 (a) is set, and while considering the element ahead located from the decomposition point as the array of Space ID "10", the element located more back than the decomposition point is considered as the array of Space ID "11."

[0048] Also in this case, CPU12 borders the array of "space ID "10" on the decomposition point through the control signal line 25. When the instruction of a purport [Space / of Space ID "10" / the array and Space ID "it decomposes into the array of 11""] is emitted, each memory module 14 Processing which carries out an abbreviation response is performed to step 1101 thru/or step 1106 of drawing 11 , and the thing (the example of drawing 18 memory module 14- 1-14 -x) relevant to an instruction performs predetermined processing among memory modules. For example, MPU36 updates the value about total size among the space management ID tables about Space ID "010" while creating the various values in the space ID managed table about Space ID "011", when the element located more back than the decomposition point is held. Moreover, also when only the element ahead located from the decomposition point is held, MPU36 of a memory module updates the value about

total size among the space management ID tables about Space ID "010." Drawing 19 is the arrays 1901 and 1902 acquired by doing in this way, and drawing showing the space ID managed table (for example, signs 1911 and 1912 and 1913 reference) in each memory module 14-1 to 14-x.

[0049] Explanation is simply added about the parallel copy which used the rearrangeable bus depending on the case [more concrete : parallel copy of multi-space memory of operation], next under multi-space memory. For example, according to the single instruction from CPU12, as shown in drawing 20 , the parallel copy of the data to other memory module groups [group / 140 / one / memory module] 141 is realizable. The following modes can be considered to a parallel copy.

[0050] (1) When a single memory module is contained in one memory module group 140 and two or more memory modules are contained in the memory module group of another side.

(2) When two or more memory modules are contained in one memory module group 140 and two or more memory modules are contained in it also at the memory module group of another side.

[0051] In the former, MPU36 of a memory module 14 which has held the element of a copied material receives the instruction (for example, command "copy the predetermined element under array which has a certain space ID as an array of space 8, 9, and ID 10") given through the control signal line 25 from CPU12, and outputs the element specified from the RAM core 34 on a predetermined bus. On the other hand, the MPU36 used as a copy place also answers acceptance of the same instruction, and while receiving the element outputted from the bus and memorizing this to the predetermined field of the RAM core 34, the space ID managed table of self is updated.

[0052] In the latter, it is possible to give the data from the memory module in one memory module group 140 to the memory module with which the memory module group 141 of another side corresponds using two or more buses, respectively. In this case, CPU12 should just control a switch 28 and a switch 30 possible [transfer of the data between predetermined memory modules].

[0053] [-- more concrete : of multi-space memory of operation -- the suffix inputted by subscript conversion is changed using the multi-space memory which hides and starts the gestalt of] book operations, such as updating, by the suffix [finishing / conversion], an array can be specified and a value can be further embellished in an array element. A certain processing is completed, and when subscript conversion and value qualification become unnecessary by committing, MPU of each memory module can cancel subscript conversion in an instant by rewriting the space ID managed table

about the array concerned, and performing RIMAPPINGU. Since it is necessary to update the element memorized by the RAM core with the value qualification actual on the other hand itself, time amount is required. Therefore, in each memory module, a changed flag is formed, and after the element with which value qualification was reflected memorizes to a RAM core actually, it is set to "1" in the flag corresponding to the element concerned. If it does in this way, when this is "1" with reference to a conversion flag in a certain process, it is not necessary to pass through value qualification, and, on the other hand, the Bayh whose conversion flag is "0" can know easily that it is necessary to pass through value qualification. Therefore, a commitment is substantially realizable in an instant. Furthermore, if the multi-space memory concerning the gestalt of this operation is used, as shown in drawing 21 , it will become possible by forming a changed flag and referring to this changed flag also about value qualification of a nested structure, to get to know the existence of the need of passing through value qualification.

[0054] [-- utilization [of multi-space memory and a rearrangeable bus]: -- in the gestalt of sort (the 1)] book operation, it becomes possible by using multi-space memory and a rearrangeable bus to perform sorting application in juxtaposition based on the single instruction from CPU12. Hereafter, explanation is added per [in the gestalt of this operation] juxtaposition-sorting application. Drawing 23 and drawing 24 are drawings for explaining the flow of the sorting application concerning the gestalt of this operation. By this sorting application, it can roughly divide, can divide into the processing (decision of the number of existence, and calculation of accumulating totals) shown in drawing 23 , and the processing (transfer of a record number) shown in drawing 24 , and can think.

[0055] In order to realize sorting application concerning the gestalt of this operation, in the gestalt of this operation, the pointer to the value list constituted so that the pointer value which shows the storing location of a corresponding value list by considering the record-number array which stored the record number, the value list which stored the actual item value about a certain item, and the value (record number) from a record-number array as an input might be outputted uses. That is, the pointer value to the corresponding value list of a location from a record number is referred to, and a actual item value is specified according to the pointer value (refer to drawing 25). First, if CPU12 gives a required instruction to each memory module 14 through the control signal line 25, processing of an abbreviation EQC will be performed by step 1101 thru/or step 1106 of drawing 11 with each memory module. Moreover, according to the advice from a memory module which stored the record number among related

memory modules, CPU12 controls switches 28 and 30 to connect to a certain bus (for "the 1st bus" to be called) the output of a series of memory modules (1st memory module group 2301) which stored the record number.

[0056] Subsequently, according to the advice from a memory module which stored the pointer array to a value list, CPU12 controls switches 28 and 30 to connect to a certain bus (for "the 2nd bus" to be called) the output of a series of memory modules (2nd memory module group 2302) which stored the pointer array to the above-mentioned value list. Furthermore, in other memory modules (3rd memory module group 2303) of a series of, the field for the "number array of existence" of the same size (the same number of elements) as the pointer to a value list is secured, and each element is initialized by "0." Furthermore, the input of the 3rd memory module group is connected with the 2nd bus of the above.

[0057] Subsequently, a record number is sent out to the 1st bus sequentially from the head of a record-number array. In the 1st memory module group 2301, MPU36 of each memory module detects the timing to which self outputs data to the 1st bus with reference to a space ID managed table, and this is realized by sending out a predetermined record number. A record number is given to each of the memory module which constitutes the 2nd memory module group 2302 through the 1st bus. MPU36 of each memory module detects that the record number relevant to the pointer array to the value list which self manages was inputted with reference to the space ID managed table of self, and outputs the pointer value corresponding to the input concerned to the 2nd bus.

[0058] Pointer value is given to each of the memory module which constitutes the joule group of the 3rd memory through the 2nd bus. MPU36 of each memory module detects that the pointer value relevant to the pointer array of the value list which self manages was given with reference to the space ID managed table of self, and increments the element of the location corresponding to pointer value in the number array of existence. By repeating this actuation, it can know what times it is pointed out to the item value by the record number (does it point?). A fixed field is secured to a series of memory modules in order to create the array which stores the sorted record number, after a series of processings for the above-mentioned number array of existence are completed. This memory module of a series of is called the 4th memory module group 2304. CPU12 controls switches 28 and 30 to connect the output of the 3rd memory module group used for previous processing, and the input of the 4th memory module group through a bus (for "the 3rd bus" to be called).

[0059] Sorting application is performed after such preparation is completed. More

specifically, a record number is given to the memory module which constitutes the 2nd memory module group through the 1st bus from the head of a record-number array. In the predetermined memory module in the 2nd memory module group, MPU36 answers acceptance of a record number and transmits pointer value to the 3rd module group through the 2nd bus. Subsequently, in a predetermined memory module, MPU36 determines the storing location of a record number with reference to the related number array of existence among the 3rd memory module group based on pointer value. Thereby, ** is sent out for a record number and its storing location to the 3rd bus from the memory module concerned. Therefore, in the predetermined memory module of the 4th memory module group, MPU36 arranges a record number in a predetermined storing location. By repeating this processing, the array (sign 2410 of drawing 24) of the record number sorted by the 4th memory module group can be created.

[0060] For example, processing shown in drawing 23 can be made into pipeline processing. That is, in the 1st bus, when a certain record number "p" is transmitted, in the 2nd bus, the pointer value "P (p-1)" about a record number "p-1" may be transmitted. Moreover, it is possible to make into pipeline processing similarly processing shown in drawing 24 . Also in this case, in the 1st bus, when a certain record number "p" is transmitted, in the 2nd bus, the pointer value "P (p-1)" about a record number "p-1" may be transmitted. Furthermore, in the 3rd bus, the storing location about a record number "p-1" may be transmitted to the same timing.

[0061] The following results were obtained about the processing time of such pipeline processing. First, the 1st bus thru/or 4th bus assumed that it was 128 bits, and thought that there is 12.8GB/second of transfer capability, respectively, and a record number and pointer value were 32 binary integers, respectively about processing of drawing 23 , respectively. In order to perform pipeline processing now although 4 billion bytes of transfer occurs in the above-mentioned processing when record count is 1 billion pieces, it turned out that it completes in 4G / 12.8G= 0.3125 seconds. Although similarly 8 billion bytes of transfer will occur when record count is 1 billion pieces if the same transfer capability and data size are assumed about processing of drawing 24 , according to the gestalt of this operation, processing can be completed in 8G / 12.8G= 0.625 seconds by activation of pipeline processing.

[0062] [-- utilization [of multi-space memory and a rearrangeable bus]: -- explanation is simply added per [by sort (the 2)], next other technique] sorting application. Also in this sorting application, the input of the 2nd memory module group 2602 which consists of a memory module which the output of the 1st memory module

group (sign 2601 reference of drawing 26) which consists of a memory module which stored the record-number array first, and the 1st bus were connected, and stored the pointer array to a value list is connected with the 1st bus. Thereby, the transfer of the output of the 1st memory module group 2601 is attained through the 1st bus at the 2nd memory module group 2602. While the field of the array which has number same on the other hand as the 2nd memory module group 2602 of the space ID is secured to the 3rd memory module group 2603, the output of the 2nd memory module group 2602 and the input of the 3rd memory module group are connected through the 2nd bus.

[0063] Subsequently, in the 1st memory module group 2601, if MPU36 of a memory module which holds a certain record number sends out the record number concerned to the 1st bus, in the predetermined memory module of the 2nd memory MOJU group 2602, MPU36 computes Space ID and sends out a record number and Space ID to the 2nd bus from the pointer value which answers and corresponds to this acceptance.

[0064] In the 3rd memory module group, based on the space ID concerned and a record number, the predetermined memory module 36 starts and the record number given to the tail of an array which has the space ID concerned is arranged. After performing such processing about all record numbers, in the 3rd memory module group, MPU36 of each memory module performs processing for combining the array which self has. High-speed sorting application is realizable with such technique.

[0065] [-- utilization [of multi-space memory and a rearrangeable bus]: -- retrieval (the 1)] -- in the gestalt of this operation, retrieval processing can be performed in juxtaposition by using multi-space memory and a rearrangeable bus again based on the single instruction from CPU12. Drawing 27 and drawing 28 are drawings for explaining the flow of the retrieval processing concerning the gestalt of this operation. For this retrieval processing, a record-number array, the propriety flag array to a value list mentioned [which mention later and value-lists / which lists and pointer-arranges] later are used. Therefore, a value is referred to in order of a record number, pointer value, and an item value like drawing 25 also in this example.

[0066] First, if CPU12 gives a required instruction to each memory module 14 through the control signal line 25, processing of an abbreviation EQC will be performed by step 1101 thru/or step 1106 of drawing 11 R> 1 with each memory module. Moreover, according to the advice from a memory module which stored the value list among related memory modules, CPU12 controls switches 28 and 30 to connect to a certain bus (for "the 1st bus" to be called) the output of a series of memory modules (1st memory module group 2701) which stored the value list. Furthermore, the field for the

propriety flag array as the thing of a value list with the same number of iodines is secured to a series of memory modules (2nd memory module group 2702), and MPU36 of each memory module belonging to the 2nd memory module 2702 concerned initializes the element of the field concerned to "0."

[0067] Subsequently, the input of the 2nd memory module group 2702 is connected to the 1st bus. Subsequently, according to the retrieval conditions given from CPU12, MPU36 sets to "1" the value to which a propriety flag array corresponds in each memory module of the 2nd memory module group with reference to the location of the item value corresponding to the retrieval conditions under value list. For example, what is necessary is just to use a split half method etc., if retrieval conditions are range. Moreover, what is necessary is just to judge the propriety for every element, if it is other conditions. A search is performed after such processing is completed. First, while connecting to the 1st bus the output of a series of memory modules (3rd memory module group 2703) which stored the record-number array, CPU12 controls switches 28 and 30 to connect to the 1st bus the input of a series of memory modules (4th memory module group 2704) which stored the pointer array to a value list. Moreover, CPU12 controls switches 28 and 30 to connect the input of the 2nd memory module group 2702 with the 2nd bus for the output of the 4th memory module group 2704.

[0068] Furthermore, the field for the array which has the same number of elements as the number of elements of a record number is secured to a series of memory modules (the 5th memory module 2705), and CPU12 controls switches 28 and 30 so that the input and the output of the 2nd memory module group 2702 are connected through the 3rd bus. After such processing, a record number is sent out to the 1st bus sequentially from the head of a record-number array. In the 3rd memory module group 2703, MPU36 of each memory module detects the timing to which self outputs data to the 1st bus with reference to a space ID managed table, and this is realized by sending out a predetermined record number.

[0069] A record number is given to each of the memory module which constitutes the 4th memory module group 2704 through the 1st bus. MPU36 of each memory module detects that the record number relevant to the pointer array to the value list which self manages was inputted with reference to the space ID managed table of self, and outputs the received record number and the pointer value corresponding to the input concerned to the 2nd bus. Pointer value is given to each of the memory module which constitutes the joule group of the 3rd memory through the 2nd bus with a record number. MPU36 of each memory module judges whether the propriety flag with which

it detects that the pointer value which shows the same location as the location of the propriety flag array which self manages was given with reference to the space ID managed table of self, and the pointer value concerned shows it is "0", or it is "1." Subsequently, when a propriety flag is "1", a related record number is given to the 5th memory module group 2705 through the 3rd bus.

[0070] In the 5th memory module group 2705, MPU36 of each memory module detects that the record number which shows the same location as the location of the array for hit information storing which self manages was given with reference to the space ID managed table of self, and sets the element of the location to "1." Retrieval is completed by taking out the element which is "1" in the array for hit information storing repeatedly about a predetermined record number about such processing.

[0071] The processing which also explained the above-mentioned retrieval processing with reference to drawing 27 , and the processing explained with reference to drawing 28 are realizable in pipeline processing like sorting application, respectively. The following results were obtained about the processing time of the pipeline processing in retrieval processing. I thought that the transfer capability of a bus and the number of bits of each element were the same as that of sorting application. In order to perform pipeline processing although 8 billion bytes of transfer occurs in the above-mentioned retrieval processing when record count is 1 billion pieces, it turned out that it completes in 8G / 12.8G= 0.624 seconds.

[0072] Furthermore, if this retrieval processing is used, the retrieval of two or more items which combined AND, OR, or NOT is also realizable. What is necessary is to take lessons from each item, to more specifically create the array for hit information storing, and just to perform logical operation between these array elements. For example, in AND of two items, or OR retrieval, a transfer (1 billion bytes) of the array element for hit information storing is performed. Therefore, the processing time can understand a required thing only for (10G/8) / 12.8G= 0.098 seconds.

[0073] In addition, what is necessary is just to connect to a column the memory module group which performs two retrieval processings in AND retrieval, in order to attain improvement in the speed further. Moreover, if an array is arranged so that the 4th memory module group and the 2nd memory module group can be constituted from two or more same memory modules, a bottleneck can be canceled and this will become possible [obtaining twice as many abbreviation / as this / processing speed].

[0074] Various modification is possible for this invention within the limits of invention indicated by the claim, without being limited to the gestalt of the above operation, and it cannot be overemphasized that it is that by which they are also included within the

limits of this invention. For example, in the gestalt of said operation, although it has applied to the computer system, it is not limited to this, and this invention can also be applied to a computer board connectable with a personal computer etc. In this case, in drawing 1 , CPU12, the memory unit 14, and bus 24 grade are carried on a board, and this constitutes the information processing unit in this invention.

[0075] Moreover, the number of the groups of the bus which connects between CPU12 and memory modules 14 and/or between memory modules 14 is not limited to the gestalt of said operation, and can be suitably determined in consideration of the magnitude of the circuit board in which a computer system is carried, the number of bits of each bus, etc. Moreover, in the gestalt of said operation, the switch 28 for specifying connection with I/O and bus of a memory module and the switch 30 from which a bus can be cut between CPU and a memory module and between memory modules are formed. forming a switch 30 -- for example, while using a certain bus (24 to bus 4 reference of drawing 1) for data transfer with the CPU module 12 and a memory module 14-1, it can be simultaneously used for the data transfer between a memory module 14-2 and a memory module 14-3 (in this case, what is necessary is just to turn OFF a switch 30-5). Therefore, it is possible to use a bus for validity more. However, when the group of a bus can enlarge a number enough, or when there are comparatively few memory modules, it is not necessary to necessarily form a switch 30.

[0076] Moreover, in this description, although it indicated that the instruction from CPU12 was given through the control signal line 25, it cannot be overemphasized that various control signals, such as a clock besides an instruction, for each memory module to synchronize and operate are given through the control signal line 25, and the predetermined signal (for example, an error signal and the signal which shows data acceptance) from each memory module to CPU12 is given. Furthermore, in this description, even if the function of one means is realized by two or more physical means, the function of two or more means may be realized by one physical means.

[0077]

[Effect of the Invention] The element under array memorized by various memory with a single instruction is outputted and inputted, and, according to this invention, it becomes possible in a distributed memory type to offer the computer architecture which can realize high-speed remarkable parallel processing.

[Translation done.]

TECHNICAL FIELD

[Industrial Application] This invention relates to the computer architecture in which general-purpose parallel operation is possible by memory control more suitable for a detail, and high-speed about the architecture of the parallel computer which can realize SIMD (Single Instruction Stream, Multiple Data Stream).

[Translation done.]

PRIOR ART

[Description of the Prior Art] A computer is introduced into various locations of the whole society, and large-scale data came to be stored here [there] by the end of today when networks including the Internet permeated. In order to process such large-scale data, huge count is required, therefore trying to introduce parallel processing is natural.

[0003] Now, parallel processing architecture is divided roughly into a "shared memory mold" and a "distributed memory type." The former ("shared memory mold") is a method with which two or more processors share one huge room. Since the traffic between a processor group and a shared memory serves as a bottleneck by this method, it is not easy to build a realistic system using the processor exceeding 100. In case the square root of 1 billion floating point variables is calculated by following, the acceleration ratio to a single CPU will call it at most 100 times. Experientially, about 30 times is an upper limit. Each processor has respectively local memory, and the latter ("distributed memory type") combines these, and builds a system. The design of the hardware system which also incorporated hundreds - tens of thousands of processors by this method is possible. Therefore, it is possible to make the acceleration ratio to the single CPU at the time of calculating the square root of the 1 billion above-mentioned floating point variables one 10,000 times the number [hundreds -] of this. However, some technical problems mentioned later exist also in

the latter. This application is made to perform the comparison with the conventional technique about a "distributed memory type", adding some considerations first about this method.

[Translation done.]

EFFECT OF THE INVENTION

[Effect of the Invention] The element under array memorized by various memory with a single instruction is outputted and inputted, and, according to this invention, it becomes possible in a distributed memory type to offer the computer architecture which can realize high-speed remarkable parallel processing.

[Translation done.]

MEANS

[Means for Solving the Problem] [-- 1st technical-problem: -- the 1st technical problem of division-of-responsibilities management] "a distributed memory type" of a huge array is a problem of division-of-responsibilities management of data. Huge data (since it is generally an array, an array explains henceforth) cannot be held in the local memory which one processor owns, and division-of-responsibilities management is inevitably carried out at two or more local memories. If an efficient and flexible division-of-responsibilities management mechanism is not introduced, it is clear that various failures will be held on the occasion of development and activation of a program.

[0005] [-- 2nd technical-problem: -- the effectiveness of interprocessor communication -- low -- as for access to the array element which other processors own, interprocessor communication is made indispensable although it can access promptly to the array element on the local memory which self owns, if each processor

of] distribution memory mold system tends to access a huge array. It is said that this interprocessor communication has extremely low performance, and is cut in 100 clocks also at the lowest compared with the communication link with a local memory. For this reason, at the time of sort implementation, since reference covering the huge array whole region is carried out and interprocessor communication occurs frequently, performance falls extremely.

[0006] Explanation is more concretely added about this trouble. 1999 current and a personal computer are constituted as a "shared memory mold" using CPU of 1 - some. It operates with an about 5 to 6 times [of a memory bus] internal clock, that interior is equipped with the automatic parallel execution function or the pipeline processing function, and standard CPU used for this personal computer can process about 1 data with one clock (memory bus). When the personal computer which is a "shared memory mold" performs sorting application of a huge array, demonstrating the multiprocessor system 100 times the performance of a "distributed memory type" which requires one clock about one data and one data takes 100 clocks (memory bus) for this reason is also considered.

[0007] The 3rd technical problem of [supply "a distributed memory type" of the technical-problem:program which is the 3rd] is a problem how to supply a program to many processors. Dramatically, a separate program is loaded to many processors and they take a great quantity of loads for creation of a program, compile, and distribution by the method (MIMD:Multiple Instruction Stream, Multiple Data Stream) which carries out coordination actuation of the whole. On the other hand, by the method (SIMD:Single Instruction Stream, Multiple Data Stream) which operates many processors by the same program, the degree of freedom of a program decreases and the situation where the program which brings about the result of a request cannot be developed is also assumed.

[0008] This invention offers the approach and computer architecture which solve the above 1st of a "distributed memory type" thru/or the technical problem of 3. The technical problem of the 1st "division-of-responsibilities management of a huge array" can solve arrangement (physical address) of each element of an array by carrying out division-of-responsibilities management by the approach with each unific processor module. By this technique, the need for a garbage collection can be lost and the tacit processing (un-explicit) assignment of each processor indispensable when insertion and deletion of an array element are completed with a number clock and realizes SIMD can also be assigned. This approach is explained like after by the concept of "multi-space memory."

[0009] the 2nd effectiveness of "interprocessor communication -- low -- " -- a technical problem can switch each interprocessor according to the processing which it be going to attain, it be the sequence that the data of the defined class be able to be defined, for every connection path, it schedule-ize a communication link so that the capacity of a bus may be use in the one direction to about 100% by carry out a continuation transfer, and it can solve it by realize huge pipeline processing simultaneously. The back will illustrate the structure-of-a-system approach which completes the sort of 1 billion lines in about 1 second by the realistic system design, in order to prove the effectiveness. This is high-speed 10,000 or more times compared with the equipment of the known maximum high speed. This approach is explained as a "recombination bus" technique like after.

[0010] The technical problem of the 3rd "supply of a program" is solvable by adopting a SIMD method. Although the tacit processing (un-explicit) assignment of each processor is determined how in the case of SIMD or ? is the biggest problem, this processing assignment can be automatically determined by the above-mentioned "multi-space memory" technique, and even if it is SIMD, the degree of freedom of a program can be held. That is, this invention outputs and inputs the element under array memorized by various memory with a single instruction, and aims at offering the computer architecture which can realize high-speed remarkable parallel processing in a distributed memory type.

[Translation done.]

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] Drawing 1 is a block diagram which shows the configuration of the computer system concerning the gestalt of operation of this invention.

[Drawing 2] Drawing 2 is a block diagram which shows the outline of the memory module concerning the gestalt of this operation.

[Drawing 3] Drawing 3 is drawing showing arrangement of a series of data in single room.

[Drawing 4] Drawing 4 is drawing showing arrangement of a series of data in the

multi-space memory concerning this invention.

[Drawing 5] Drawing 5 is drawing for explaining ADORESURI mapping in the gestalt of this operation.

[Drawing 6] Drawing 6 is drawing for explaining the value qualification in the gestalt of this operation.

[Drawing 7] Drawing 7 is drawing showing the outline of the pipeline processing between the memory modules concerning the gestalt of this operation.

[Drawing 8] Drawing 8 is drawing for explaining the structure of a memory module 14 under the multi-space memory concerning the gestalt of this operation.

[Drawing 9] Drawing 9 is drawing for explaining the structure of a memory module 14 under multi-space memory.

[Drawing 10] Drawing 10 is drawing for explaining the structure of a memory module 14 under multi-space memory.

[Drawing 11] Drawing 11 is a flow chart which shows the processing performed with each memory module which received the instruction of deleting the element of the predetermined range in a certain space ID.

[Drawing 12] Drawing 12 is drawing showing relation with arrangement of the element deleted and the element currently held with the memory module.

[Drawing 13] Drawing 13 is a flow chart which shows the processing performed with each memory module which received the instruction of deleting the element of the predetermined range in a certain space ID.

[Drawing 14] Drawing 14 is a flow chart which shows the processing performed with each memory module which received the instruction of adding an element to the tail of the array of a certain space ID.

[Drawing 15] Drawing 15 is drawing for explaining association of the array concerning the gestalt of this operation, and division of an array.

[Drawing 16] Drawing 16 is drawing showing the condition that these were held into the memory module by the array and list which have the array which has Space ID "10", and Space ID "11" in the gestalt of this operation.

[Drawing 17] Drawing 17 is the array acquired by association of an array, and drawing showing the space ID managed table in each memory module in the gestalt of this operation.

[Drawing 18] Drawing 18 is drawing showing an example divided into the array which has Space ID "10" for the array which has Space ID "10", and the array which has Space ID "11" in the gestalt of this operation.

[Drawing 19] Drawing 19 is the array acquired by division of an array, and drawing

showing the space ID managed table in each memory module in the gestalt of this operation.

[Drawing 20] Drawing 20 is drawing showing the parallel copy of the gestalt of this operation, or the data to other memory module groups [group / one / to cut / memory module].

[Drawing 21] Drawing 21 is drawing for explaining utilization of the changed flag concerning the gestalt of this operation.

[Drawing 22] Drawing 22 is drawing for explaining utilization of the changed flag concerning the gestalt of this operation.

[Drawing 23] Drawing 23 is drawing for explaining the flow of the sorting application concerning the gestalt of this operation.

[Drawing 24] Drawing 24 is drawing for explaining the flow of the sorting application concerning the gestalt of this operation.

[Drawing 25] Drawing 25 is drawing showing the reference procedure of data until an item value is specified from a record number in the gestalt of this operation.

[Drawing 26] Drawing 26 is drawing for explaining the flow of other sorting application concerning the gestalt of this operation.

[Drawing 27] Drawing 27 is drawing for explaining the flow of the retrieval processing concerning the gestalt of this operation.

[Drawing 28] Drawing 28 is drawing for explaining the flow of the retrieval processing concerning the gestalt of this operation.

[Description of Notations]

10 Computer System

12 CPU Module

14 Memory Module

16 Fixed Memory

18 Input Unit

20 Display

22 Legacy Memory

24 Bus

25 Control Signal Line

26 Bus

28 30 Switch

32 Clock Buffer

34 RAM Core

36 MPU

38 I/O

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号
特開2001－92796
（P2001－92796A）
(43)公開日　平成13年４月６日(2001.4.6)

(51)Int.Cl.⁷		識別記号	FI		テーマコード（参考）
Ｇ０６Ｆ	15/167		Ｇ０６Ｆ 15/167	Ｂ	５Ｂ００５
	12/02	５１０	12/02	５１０Ａ	５Ｂ０４５
	12/06	５３０	12/06	５３０Ｄ	５Ｂ０６０
	12/08		12/08	Ｈ	
	12/10		12/10	Ｈ	

審査請求　未請求　請求項の数11　ＯＬ　（全 23 頁）　最終頁に続く

(21)出願番号　特願平11－263793

(22)出願日　平成11年９月17日(1999.9.17)

(71)出願人　599074648
ターボデータラボラトリー有限会社
東京都台東区松が谷１－９－12　ＳＰＫビルディング604号
(72)発明者　古庄　晋二
神奈川県横浜市神奈川区松見町４丁目1101番地７コートハウス菊名804号
(74)代理人　100103632
弁理士　窪田　英一郎　（外１名）

Ｆターム(参考)　5B005 KK14 KK22 MM32
5B045 BB12 CG14
5B060 MB04

(54)【発明の名称】　並列コンピュータのアーキテクチャおよびこのアーキテクチャを利用した情報処理ユニット

(57)【要約】
【課題】　分散メモリー型において、著しく高速な並列処理を実現可能なコンピュータアーキテクチャを提供する
【解決手段】　コンピュータシステム１０は、ＣＰＵモジュール１２と、それぞれがＭＰＵ３６およびＲＡＭコア３４とを有する複数のメモリモジュール１４と、ＣＰＵとメモリモジュールとの接続やメモリモジュール間の接続をなす複数組のバス２４とを備え、ＣＰＵ１２から与えられるインストラクションにより、各メモリモジュールが作動する。所定の関連を有する一連のデータには、空間ＩＤが付与され、各メモリモジュールが、少なくとも、当該空間ＩＤ、自己が管理する一連のデータの部分に関する論理アドレス、一連のデータのサイズを含むテーブルを管理し、かつ、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータに関する処理を実行する。

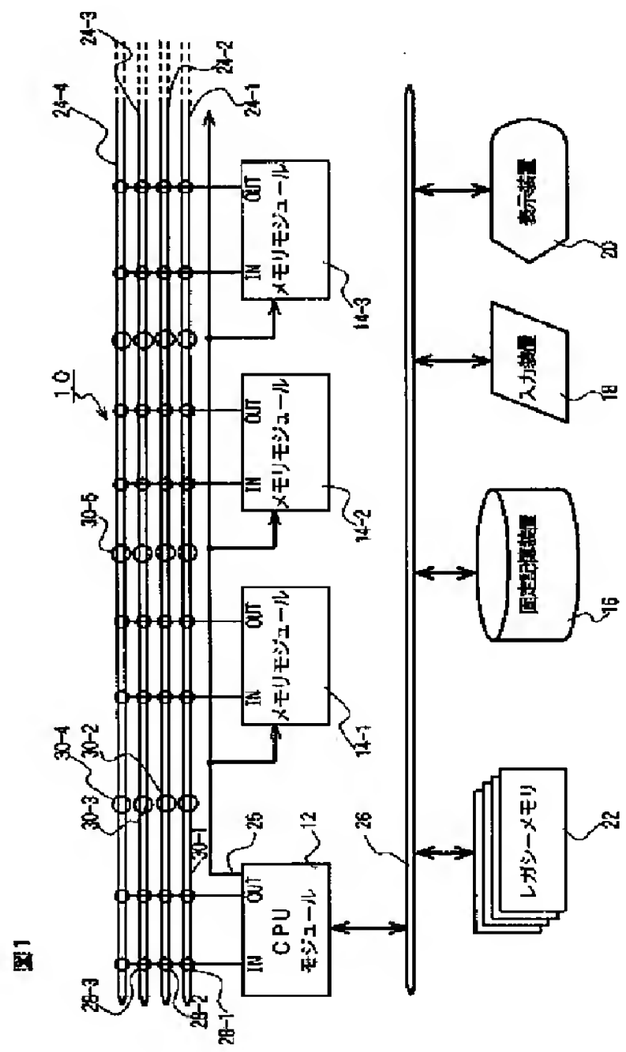


図1

1

【特許請求の範囲】

【請求項１】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵが作動するように構成された並列コンピュータのアーキテクチャであって、

所定の関連を有する一連のデータに、空間ＩＤが付与され、各メモリモジュールのＭＰＵが、少なくとも、当該空間ＩＤ、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、

各メモリモジュールのＭＰＵが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをＲＡＭコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする並列コンピュータのアーキテクチャ。

【請求項２】　前記ＭＰＵが、ＣＰＵから与えられた空間ＩＤを、自己が管理する１以上の一連のデータの空間ＩＤと比較する空間コンパレータと、

ＣＰＵから与えられた論理アドレスと、自己が管理するデータの部分の論理アドレスとを比較するアドレスコンパレータと、

当該論理アドレスに基づき、自己のＲＡＭセル上の物理アドレスを算出するアドレスカリキュレータとを有することを特徴とする請求項１に記載のコンピュータアーキテクチャ。

【請求項３】　前記メモリモジュールの各々が、ＣＰＵモジュールおよび他のメモリモジュールとの同期をなすための同期信号を受け入れ、かつ、

前記複数組のバスの何れかとの接続が可能な入力と、前記複数組のバスの他の何れかとの接続が可能な出力を備え、少なくとも、前記同期信号にしたがって、前記何れかのバスと入力との接続により、データを入力しつつ、前記他の何れかのバスと出力との接続により、データを出力できるように構成されたことを特徴とする請求項１または２に記載のコンピュータアーキテクチャ。

【請求項４】　前記複数組のバスの各々に、前記ＣＰＵモジュールと何れかのメモリモジュールの入力または出力との間、および／または、他の何れかのメモリモジュールの入力または出力と、さらに他のメモリモジュールの出力または入力との間の接続を規定するためのスイッチが設けられ、

前記スイッチの切換により、複数組のバスの各々において、並列的にデータの授受が実現されることを特徴とする請求項３に記載のコンピュータアーキテクチャ。

2

【請求項５】　前記複数組のバスのうちの何れかである第１のバスに、何れかのメモリモジュールの出力と、他の何れかのメモリモジュールの入力とが接続され、かつ、前記複数組のバスのうち、他の何れかである第２のバスに、当該他の何れかのメモリモジュールの出力と、さらに他の何れかのメモリモジュールの入力とが接続され、第１のバスにおけるデータの授受と、第２のバスにおけるデータの授受が並列的に進行することを特徴とする請求項４に記載のコンピュータアーキテクチャ。

【請求項６】　前記バスとメモリモジュールとの間の接続を繰り返して、多段のメモリモジュール間の接続を形成することを特徴とする請求項５に記載のコンピュータアーキテクチャ。

【請求項７】　前記ＭＰＵが、一連のデータ中の特定の要素を削除し、前記一連のデータ中に特定の要素を挿入し、或いは、一連のデータの末尾に特定の要素を追加することを示すインストラクションを受理すると、テーブルを参照して、自己の管理するデータの領域と、削除、挿入或いは追加にかかる要素の位置とを比較して、当該比較結果に応じて、前記テーブルの内容を更新することを特徴とする請求項１ないし６の何れか一項に記載のコンピュータアーキテクチャ。

【請求項８】　前記ＭＰＵが、与えられたインストラクションに応答して、一連のデータ中の要素を特定するための添え字を変換し、および／または、要素に特定の修飾を与える値変換を実行することを特徴とする請求項１ないし７の何れか一項に記載のコンピュータアーキテクチャ。

【請求項９】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵが作動するように構成された情報処理ユニットであって、

所定の関連を有する一連のデータに、空間ＩＤが付与され、各メモリモジュールのＭＰＵが、少なくとも、当該空間ＩＤ、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、

各メモリモジュールのＭＰＵが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをＲＡＭコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする情報処理ユニット。

【請求項１０】　前記ＣＰＵモジュールが、レガシーメ

モリ、入力装置および表示装置を相互接続する他のバスと連結可能に構成されたことを特徴とする請求項９に記載の情報処理ユニット。

【請求項１１】　請求項９に記載の情報処理ユニットと、ＣＰＵモジュールと他のバスを介して連結された１以上のレガシーメモリを含む記憶装置、入力装置および表示装置とを有することを特徴とするコンピュータシステム。

【発明の詳細な説明】

【０００１】

【産業上の技術分野】本発明は、ＳＩＭＤ(Single Instruction Stream, Multiple Data Stream)を実現可能な並列コンピュータのアーキテクチャに関し、より詳細には、適切かつ高速なメモリ制御により、汎用的な並列演算が可能なコンピュータアーキテクチャに関する。

【０００２】

【従来の技術】社会全体のさまざまな場所にコンピュータが導入され、インターネットをはじめとするネットワークが浸透した今日では、そこここで、大規模なデータが蓄積されるようになった。このような大規模データを処理するには、膨大な計算が必要で、そのために並列処理を導入しようと試みるのは自然である。

【０００３】さて、並列処理アーキテクチャは「共有メモリ型」と「分散メモリ型」に大別される。前者（「共有メモリ型」）は、複数のプロセッサが１つの巨大なメモリ空間を共有する方式である。この方式では、プロセッサ群と共有メモリ間のトラフィックがボトルネックとなるので、百を越えるプロセッサを用いて現実的なシステムを構築することは容易ではない。したがって、例えば１０億個の浮動小数点変数の平方根を計算する際、単一ＣＰＵに対する加速比は、せいぜい１００倍ということになる。経験的には、３０倍程度が上限である。後者（「分散メモリ型」）は、各プロセッサがそれぞれローカルなメモリを持ち、これらを結合してシステムを構築する。この方式では、数百～数万ものプロセッサを組み込んだハードウェアシステムの設計が可能である。したがって、上記１０億個の浮動小数点変数の平方根を計算する際の単一ＣＰＵに対する加速比を、数百～数万倍とすることが可能である。しかしながら、後者においても、後述するいくつかの課題が存在する。本出願は、「分散メモリ型」に関するものであり、この方式について最初に多少の考察を加えながら従来技術との比較を行うことにする。

【０００４】

【課題を解決するための手段】[第１の課題：巨大配列の分掌管理]「分散メモリ型」の第１の課題は、データの分掌管理の問題である。巨大なデータ（一般的には配列なので、以降、配列で説明する）は、１つのプロセッサの所有するローカルメモリに収容できるものではなく、必然的に複数のローカルメモリに分掌管理される。

効率的かつ柔軟な分掌管理メカニズムを導入しないと、プログラムの開発および実行に際してさまざまな障害を抱え込むことになることは明らかである。

【０００５】[第２の課題：プロセッサ間通信の効率の低さ]分散メモリ型システムの各プロセッサが、巨大配列にアクセスしようとすると、自己の所有するローカルメモリ上の配列要素に対しては速やかにアクセスできるものの、他のプロセッサが所有する配列要素へのアクセスはプロセッサ間通信を必須とする。このプロセッサ間通信はローカルメモリとの通信に比べ、極端にパフォーマンスが低く、最低でも１００クロックかかると言われている。このため、ソート実施時には、巨大配列全域にわたる参照が実施され、プロセッサ間通信が多発するため、パフォーマンスが極端に低下する。

【０００６】この問題点につき、より具体的に説明を加える。１９９９年現在、パソコンは、１～数個のＣＰＵを用いて、「共有メモリ型」として構成されている。このパソコンに使用される標準的なＣＰＵは、メモリバスの５～６倍程度の内部クロックで動作し、その内部に自動的な並列実行機能やパイプライン処理機能が装備されており、およそ１データを１クロック（メモリバス）で処理できる。「共有メモリ型」であるパソコンにて巨大配列のソート処理を行う場合、１データについて１クロックを要し、このため、１データに１００クロック（メモリバス）を要する、「分散メモリ型」のマルチプロセッサシステムの１００倍のパフォーマンスを発揮することも考えられる。

【０００７】[第３の課題：プログラムの供給]「分散メモリ型」の第３の課題は、多数のプロセッサにどうやってプログラムを供給するか、という問題である。非常に多数のプロセッサに、別々のプログラムをロードし、全体を協調動作させる方式（ＭＩＭＤ：Multiple Instruction Stream, Multiple Data Stream）では、プログラムの作成、コンパイル、配信のために多大な負荷を要する。その一方、多数のプロセッサを同一のプログラムで動作させる方式（ＳＩＭＤ：Single Instruction Stream, Multiple Data Stream）では、プログラムの自由度が減少し、所望の結果をもたらすプログラムが開発できない事態も想定される。

【０００８】本発明は、「分散メモリ型」の上記第１ないし３の課題を解決する方法およびコンピュータアーキテクチャを提供する。第１の「巨大配列の分掌管理」の課題は、配列の各要素の配置（物理アドレス）を、各プロセッサモジュールが統一的な方法で分掌管理することで解決できる。この手法により、ガーベージコレクションの必要性が無くなり、配列要素の挿入・削除が数クロックで完了し、ＳＩＭＤを実現する上で欠かせない各プロセッサの暗黙の（非明示的）処理分担を割り付けることもできる。この方法は、後ほど「多空間メモリ」という概念で説明される。

5

【０００９】第２の「プロセッサ間通信の効率の低さ」の課題は、達成しようとする処理に応じて各プロセッサ間をつなぎ替え、各接続経路毎に、定められた種類のデータを、定められた順番で、１方向に連続転送することで、バスの能力を１００％近くまで使用できるよう通信をスケジュール化し、同時に巨大パイプライン処理を実現することで解決できる。その有効性を実証するため、後ほど、現実的なシステム設計で、１０億行のソートを１秒程度で完了するシステムの構成方法を例示するであろう。これは、既知の最高速の装置に比べて、１万倍以上高速である。この方法は、後ほど「組替えバス」技術として説明される。

【００１０】第３の「プログラムの供給」の課題は、ＳＩＭＤ方式を採用することで解決できる。ＳＩＭＤの場合は、各プロセッサの暗黙の（非明示的）処理分担をどうやって決定するか？が最大の問題であるが、前述の「多空間メモリ」技術にてこの処理分担が自動的に決定でき、ＳＩＭＤであってもプログラムの自由度を保持することができる。つまり、本発明は、分散メモリー型において、単一命令により種々のメモリーに記憶された配列中の要素を入出力し、著しく高速な並列処理を実現可能なコンピュータアーキテクチャを提供することを目的とする。

【００１１】

【課題を解決するための手段】本発明の目的は、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵが作動するように構成された並列コンピュータのアーキテクチャであって、所定の関連を有する一連のデータに、空間ＩＤが付与され、各メモリモジュールのＭＰＵが、少なくとも、当該空間ＩＤ、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、各メモリモジュールのＭＰＵが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをＲＡＭコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする並列コンピュータのアーキテクチャにより達成される。

【００１２】本発明によれば、空間ＩＤを用いて一連のデータを把握するため、当該一連のデータが、多数のメモリモジュールにより分掌されても、各メモリモジュールのＭＰＵが、当該一連のデータを確実に認識することができる。また、メモリモジュールは、一連のデータおよび自己が管理するその部分を、テーブルにて把握しているため、インストラクションの受理にしたがって、そのテーブルを参照して、所定の処理を実行することができる。これにより、単一インストラクションに基づく、各ＭＰＵでの並列処理が実現できる。

6

【００１３】本発明の好ましい実施態様においては、ＭＰＵは、ＣＰＵから与えられた空間ＩＤを、自己が管理する１以上の一連のデータの空間ＩＤと比較する空間コンパレータと、ＣＰＵから与えられた論理アドレスと、自己が管理するデータの部分の論理アドレスとを比較するアドレスコンパレータと、当該論理アドレスに基づき、自己のＲＡＭセル上の物理アドレスを算出するアドレスカリキュレータとを有している。これらコンパレータおよびカリキュレータは、ハードウェアにて構成されても良いし、ＭＰＵのプログラムによりソフトウェアとして実現されるものであっても良い。

【００１４】また、本発明の好ましい実施態様においては、メモリモジュールの各々が、ＣＰＵモジュールおよび他のメモリモジュールとの同期をなすための同期信号を受け入れ、かつ、前記複数組のバスの何れかとの接続が可能な入力と、前記複数組のバスの他の何れかとの接続が可能な出力を備え、少なくとも、前記同期信号にしたがって、前記何れかのバスと入力との接続により、データを入力しつつ、前記他の何れかのバスと出力との接続により、データを出力できるように構成されている。本実施の形態によれば、同期信号にしたがって、メモリモジュールからのデータ出力およびメモリモジュールへのデータ入力がなされ、かつ、バスの接続の制御により、適切に並列処理を実現することが可能となる。

【００１５】複数組のバスの各々には、前記ＣＰＵモジュールと何れかのメモリモジュールの入力または出力との間、および／または、他の何れかのメモリモジュールの入力または出力と、さらに他のメモリモジュールの出力または入力との間の接続を規定するためのスイッチが設けられ、スイッチの切換により、複数組のバスの各々において、並列的にデータの授受が実現されるのがより好ましい。これにより、複数組のバスをより有効に利用することが可能となり、より並列性を高めることが可能となる。

【００１６】本発明のさらに好ましい実施態様においては、複数組のバスのうちの何れかである第１のバスに、何れかのメモリモジュールの出力と、他の何れかのメモリモジュールの入力とが接続され、かつ、前記複数組のバスのうち、他の何れかである第２のバスに、当該他の何れかのメモリモジュールの出力と、さらに他の何れかのメモリモジュールの入力とが接続され、第１のバスにおけるデータの授受と、第２のバスにおけるデータの授受が並列的に進行する。このように、コンピュータの実施態様によれば、ＣＰＵモジュールと、メモリモジュールとにより、パイプライン処理を実現することが可能と

なる。バスとメモリモジュールとの間の接続を繰り返して、多段のメモリモジュール間の接続を形成するのがより好ましい。

【００１７】本発明の別の好ましい実施態様においては、ＭＰＵが、一連のデータ中の特定の要素を削除し、前記一連のデータ中に特定の要素を挿入し、或いは、一連のデータの末尾に特定の要素を追加することを示すインストラクションを受理すると、テーブルを参照して、自己の管理するデータの領域と、削除、挿入或いは追加にかかる要素の位置とを比較して、当該比較結果に応じて、前記テーブルの内容を更新する。すなわち、ＭＰＵにおいて、自己が管理するテーブルを更新する、すなわち、リマッピングをすることにより、要素の削除、挿入および追加を実現することが可能となる。

【００１８】本発明のさらに別の実施態様においては、ＭＰＵが、与えられたインストラクションに応答して、一連のデータ中の要素を特定するための添え字を変換し、および／または、要素に特定の修飾を与える値変換を実行する。また、本発明の目的は、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、メモリモジュール間の接続をなす複数組のバスとを備え、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵが作動するように構成された情報処理ユニットであって、所定の関連を有する一連のデータに、空間ＩＤが付与され、各メモリモジュールのＭＰＵが、少なくとも、当該空間ＩＤ、自己が管理する一連のデータの部分に関する論理アドレス、当該部分のサイズ、および、一連のデータのサイズを含むテーブルを管理し、かつ、各メモリモジュールのＭＰＵが、受理したインストラクションに、自己が管理する一連のデータの部分が関与しているか否かを判断して、ＲＡＭコアに記憶されたデータを読み出してバスに送出し、バスを介して与えられたデータをＲＡＭコアに書き込み、データに必要な処理を施し、および／または、前記テーブルを更新するように構成されたことを特徴とする情報処理ユニットによっても達成される。たとえば、前記ユニットが単一の回路基板に形成され、ＣＰＵモジュールが、レガシーメモリ、入力装置および表示装置を相互接続する他のバスと連結可能に構成されていても良い。

【００１９】さらに、本発明の目的は、上記情報処理ユニットと、ＣＰＵモジュールと他のバスを介して連結された１以上のレガシーメモリを含む記憶装置、入力装置および表示装置とを有することを特徴とするコンピュータシステムによっても達成される。

【００２０】

【発明の実施の形態】［ハードウェア構成］以下、添付図面を参照して、本発明の実施の形態につき説明を加える。図１は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイヤグラムである。図１に示すように、コンピュータシステム１０は、単一命令による並列演算を実現するＣＰＵモジュール１２と、並列演算のために必要な種々のデータを記憶するメモリモジュール１４－１、１４－２、１４－３、…と、必要なプログラムやデータを記憶する固定記憶装置１６と、キーボードやマウスなどの入力装置１８と、ＣＲＴなどからなる表示装置２０と、種々の形式のデータ等が記憶されているレガシーメモリ２２とを備えている。また、バス２４－１、２４－２、…において、ＣＰＵモジュール１２、各メモリモジュール１４との接点には、スイッチ２８－１、２８－２、２８－３、…などが配設され、選択された回路要素間における情報の授受が可能となっている。また、ＣＰＵモジュール１２とメモリモジュール１４－１との間、隣接するメモリモジュール間において、バスの連結および接続をなすためのスイッチ３０－１、３０－２、…が設けられている。

【００２１】ＣＰＵモジュール１２と、メモリモジュール１４との間には、複数のバス２４－１、２４－２、２４－３、２４－４、…とが設けられている。したがって、ＣＰＵモジュール１２とメモリモジュール１４との間、および、メモリモジュール間は、上記バスによりデータ等の授受が可能となっている。また、ＣＰＵ１２と、メモリモジュール１４との間には、制御信号ライン２５が設けられ、ＣＰＵ１２から発せられるインストラクションなどが、全てのメモリモジュール１４に伝達されるようになっている。

【００２２】さらに、ＣＰＵ１２と、他の構成要素（たとえば、固定記憶装置１６、入力装置１８など）との間には、ローカルバス２６が配設されており、これらの間でもデータ等の授受が可能となっている。ＣＰＵ１２は、固定記憶装置１６に記憶され、或いは、バス２６上に接続されたＲＡＭのような他の記憶装置（図示せず）に記憶されたプログラムを読み出し、このプログラムにしたがって、以下に示すメモリモジュール１４へのインストラクションの送出を含むデータの授受のほか、スイッチ２８、３０の制御等を実行する。また、ＣＰＵ１２は、プログラムにしたがって、レガシーメモリ２２に記憶された種々の形式のデータを受け入れて、この形式のデータを、ＣＰＵ１２、メモリモジュール１４、バス２４からなる系にて処理可能な一連のデータ（配列）に変換し、これらを、各メモリモジュール１４に記憶させることもできる。

【００２３】図２は、各メモリモジュール１４の概略を示すブロックダイヤグラムである。図２に示すように、メモリモジュール１４は、ＣＰＵモジュール１２から与えられるクロックなど同期信号を受け入れるクロックバッファ３２と、データを記憶するＲＡＭコア３４と、後述する空間ＩＤやデータの要素番号等を把握し、ＣＰＵ

１２からのインストラクションなどを受理した場合に、空間ＩＤや要素番号に基づき、ＲＡＭコア３４へのデータ書き込みやＲＡＭコアからのデータ読み出しを制御するＭＰＵ３６と、バスの何れかからのデータを受け入れて、ＲＡＭコア３４に供給し、および／または、ＲＡＭコア３４からのデータを何れかのバスに送出するＩ／Ｏ３８とを有している。この実施の形態において、メモリモジュール１４は、制御信号ライン２５を介して、ＣＰＵからのインストラクションを受け入れ、ＭＰＵ３６が、このインストラクションに応答して、ＲＡＭコア３４のデータを読み出し、ＲＡＭコア３４にデータを書き込み、或いは、データに所定の処理を施すことができるようになっている。また、ＲＡＭコア３４へのデータアクセスや、Ｉ／Ｏを介してデータ入力およびデータ出力は、クロックバッファ３２に与えられるクロックなどの同期信号に基づき実行される。

【００２４】図１および図２から明らかなように、本発明において、コンピュータシステム１０は、メモリ共有型のシステムであると考えることができる。また、後述するように、制御信号ライン２５を介して、各メモリモジュール１４にインストラクションを与えることにより、各メモリモジュール１４が並列的に処理を実行する。また、バスへのデータ出力およびバスからのデータ入力などが、所定の同期信号に基づき実行される。したがって、このコンピュータシステム１０は、ＳＩＭＤの形態をなしていると考えることができる。

【００２５】［実現される機能の概略］このような構成を有するコンピュータシステム１０につきより詳細な説明を加える前に、本コンピュータシステム１０により実現される機能の概略を簡単に説明する。

（１）多空間メモリ

本明細書において、多空間メモリとは、メモリ空間を、空間ＩＤとアドレスとに基づきアクセスするために割り当てられたメモリ空間をいう。これにより、一連のデータが多数のプロセッサに分掌されていても、各プロセッサが、これを確実に分離、認識することができる。従来のメモリ空間においては、プロセス毎に個別の領域を割り当てることはあっても、一連の変数（配列、構造体など）毎に目盛り空間を割り当てることは行われてこなかった。したがって、以下、このような従来のメモリ空間を「単一メモリ空間」と称する。単一メモリ空間のシステムにおいては、アドレスのみを用いてデータにアクセスしているため、関連を有する一連のデータを分離したり、認識することができなかった。このため、実際には並列処理が可能であっても、その可否を判断できない場合が多かった。また、ある単一メモリ空間に、新たな一連のデータを収容させる場合に、当該一連のデータの収容場所を確保するために、ガーベージコレクションを実行する必要があった。

【００２６】これに対して、本発明においては、メモリ空間に、空間ＩＤを導入し、一連のデータについて同一のＩＤを付与している。また、メモリモジュール１４において、自身のＲＡＭコア３４に保持されているデータに関する空間ＩＤを把握し、これにより、各メモリモジュール１４自体が、現在アクセスされているデータの空間ＩＤを参照することにより、自己の作動の是非を決定することができる。また、各メモリモジュールが空間ＩＤと関連付けて、一連のデータの全部或いは一部を保持できるため、ある一連のデータを、複数のメモリモジュール１４に分割して記憶させることができ、これによりガーベージコレクションを不要にすることができる。

【００２７】たとえば、図３に示すように、単一メモリ空間において、“Ａ”という一連のデータ、“Ｂ”という一連のデータ、…が収容されている場合を考える。たとえば、ここで、全メモリサイズが３２ワードで、上記一連のデータのサイズの総和が３０ワードであると仮定する。これら一連のデータは、空間中に点在しているため、未使用のメモリサイズは、１２ワードであるにもかかわらず、実際に格納できる一連のデータのサイズは３ワードに限定される。このため、３ワードを超えたサイズを有する新たな一連のデータを収容すべき場合には、ガーベージコレクションを実行しなければならない。その一方、図４に示すように、本発明においては、一連のデータの各々に、空間ＩＤが付与されている。これらは、空間ＩＤと関連付けられて、１以上のメモリモジュール１４に記憶される。したがって、未使用のサイズと収容可能なサイズとを一致させることが可能となる。

【００２８】（２）メモリモジュール

また、本発明においては、各メモリモジュール１４が、ＭＰＵ３６を有し、上記空間ＩＤのほか、自己が保持する一連のデータの各々の要素番号を把握している。したがって、ＣＰＵ１２からのインストラクションを受理した後、ＭＰＵ３６が、インストラクションにしたがってアクセスすべきデータが、自己のＲＡＭコア３４中に保持されているものか否かを判断して、アクセスに必要の是非を決定することができる。さらに、各メモリモジュール１４が、自己のＲＡＭコア３４に格納されている配列要素の添え字の範囲から、ＳＩＭＤでのインストラクションにおける暗黙の処理の分担範囲を決定することが可能である。

【００２９】また、本発明においては、メモリモジュール１４が、アドレスリマッピングを実行できるようになっている。たとえば、図５に示すように、ある配列の所定の位置に特定の要素を挿入する場合、その他、所定の位置の要素を削除し、或いは、配列の末尾に所定の要素を追加する場合にも、本実施の形態においては、当該配列に関連する要素を保持しているメモリモジュールの各々において、ＭＰＵ３６が、アドレスリマッピングを実行することにより、並列的かつ高速に、これらを実現することができる。さらに、図６に示すように、配列の要

素（値）に修飾を与える場合（たとえば、各値に「１」を加える場合）にも、関連する配列の要素を保持するメモリモジュールの各々において、ＭＰＵ３６が、並列的かつ高速に、必要な処理を行うことができる。

【００３０】また、メモリモジュール１４においては、ＭＰＵ３６が、ＲＡＭコア３４にて記憶すべきデータの各々のサイズを把握し、圧縮した形態にてこれらを記憶することができる。たとえば、あるメモリモジュール１４にて、整数値のデータを保持すべき場合に、実際のデータ値が“０”ないし“３”までの値しか取り得ない場合には、ＭＰＵ３６は、各データのために２ビットのみを用意する。ＣＰＵ１２との間では、１つの整数を表現するために３２ビットを使用していた場合には、メモリモジュール１４とＣＰＵ１２との間での通信のために、ＭＰＵ３６が、データ形式を変更して、ＣＰＵ１２との授受をなせば良い。これにより、ＲＡＭコア３４をより無駄なく利用することが可能となる。また、文字列のような長さの異なるデータについても、同様にデータ長を変更して記憶することができるようになっている。

【００３１】さらに、メモリモジュール１４においては、所定の空間ＩＤに関連付けられたデータや、所定の範囲の要素番号を付されたデータに、特定の値（たとえば、「０」）をセットすることができるようになっている。これにより、メモリモジュール１４内で、高速に初期化の処理を実行することが可能となる。また、メモリモジュール１４においては、ある特定のデータ（配列）中の値を検索することや、添字の範囲をチェックすることが可能である。

【００３２】（３）組み替え可能バス

本発明においては、ＣＰＵ１２が、スイッチ２８－１、２８－２、…およびスイッチ３０－１、３０－２、…を選択的にオン／オフして、データの授受をなすべきメモリモジュール１４を指定することにより、パイプライン処理を実現している。たとえば、図７に示すように、あるメモリモジュール１４－ｉから出力されたデータを、他のメモリモジュール１４－ｊに与え、かつ、当該他のメモリモジュール１４－ｊから出力されたデータを、さらに他のメモリモジュール１４－ｋに伝達すべき場合には、ＣＰＵ１２は、バス２４－ｍを、メモリモジュール１４－ｉ、１４－ｊのために割り当て、かつ、バス２４－ｎを、メモリモジュール１４－ｊ、１４－ｋのために割り当てるように、各スイッチの状態を設定する。

【００３３】さらに、これらパイプライン処理は、単一のメモリモジュール間の接続により実現される場合だけでなく、複数の一連のメモリモジュール（メモリモジュール群）の間の接続により実現することも可能である。達成しようとする処理に応じて、各メモリモジュール間をつなぎ替え、各接続経路毎に、定められた種類のデータを定められた順序にて一方向に連続転送することで、バスの能力を１００％近く使用できるように、通信をスケジュール化することができる。これにより、分散メモリ型の並列処理システムの最大の問題であった、プロセッサ間通信のパフォーマンスの低さを、解消することができる。このように構成されたコンピュータシステム１０において、多空間メモリの具体的構成および多空間メモリにおけるシステムの作動につき説明を加える。

【００３４】［多空間メモリ］図８は、多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。図８（ａ）に示すように、メモリモジュール１４中のＲＡＭコア３４には、空間ＩＤ管理テーブルが設けられる。これにより、メモリモジュール１４のＭＰＵ３６は、自己が保持するデータの空間ＩＤ等必要な情報を把握することが可能となる。図８（ｂ）に示すように、空間ＩＤ管理テーブルには、自己が保持するデータ群ごとの、空間ＩＤ、ＣＰＵの管理の下での、データ群の論理開始アドレス、データ群が割り付けられた領域のサイズ、ＲＡＭコア３４中の物理開始アドレス、当該空間ＩＤを有する一連のデータの全サイズ、および、アクセス制限を示すアクセス制限フラグが格納されている。アクセス制限フラグは、この実施の形態においては、読み出しのみ可能（Ｒ）、書き込みのみ可能（Ｒ）、読み書き可能（ＲＷ）の３つの状態を示すことができるようになっている。

【００３５】メモリモジュール１４のＭＰＵ３６は、ある空間ＩＤを有するデータ群が与えられた際に、ＲＡＭコア３４中に当該データ群を収容すべき、１以上の領域を見出して、当該領域にデータ群をそのまま、或いは、２以上に分割して収容する。この際に、与えられた空間ＩＤ、論理開始アドレス、全サイズ、アクセス制限フラグとともに、実際にデータを収容したＲＡＭコア中の論理開始アドレスや、割り付け領域サイズも、空間ＩＤ管理テーブルに記憶される。図８（ｃ）は、図８（ｂ）による空間ＩＤ管理テーブルにしたがったＲＡＭコア３６中のデータを示す図である。

【００３６】［メモリアクセスの概略説明］このように構成されたメモリモジュール１４へのアクセスにつき以下に説明を加える。図９に示すように、まず、ＣＰＵ１２が、空間ＩＤおよび論理アドレス、並びに、必要なインストラクション（たとえば、データの書き込みや読み出し）を、制御信号ライン２５を介して、全てのメモリモジュール１４に伝達する。各メモリモジュール１４においては、これに応答して、ＭＰＵ３６に設けられた空間コンパレータ５２が、空間ＩＤと、自己の空間ＩＤ管理テーブル上に保持されている空間ＩＤとを比較して、同一のものを、自己が保持しているかを判断し、また、、アドレスコンパレータ５４が、論理アドレスについて、同様の判断を行う。次いで、メモリモジュール１４のＭＰＵ３６が、自己のＲＡＭコア３４に、インストラクションによる処理対象となるデータが保持されていると判断した場合には、アドレスカリキュレータ５６

が、空間ＩＤ管理テーブルを参照して、ＲＡＭコア３４中の物理アドレスを算出し、処理対象となるデータを特定する。このようにして、データが特定された後に、ＭＰＵ３６は、ＣＰＵ１２から与えられたインストラクションに応じた処理（たとえば、データの書き込みや読み出し）を実行し、必要な場合には、データをＣＰＵ１２に伝達する（図９（ｃ）参照）。

【００３７】［多空間メモリのより具体的な動作：配列中の要素の削除等］たとえば、ある空間ＩＤをもつ一連のデータ（以下、これを場合によって「配列」と称する。）が、１以上のメモリモジュール１４に収容された状態から、特定の要素が削除された状態までの一連の動作につき以下に説明する。あるメモリモジュール１４－ｉにおいて、空間ＩＤ「０１０」に属するデータ群が、図１０（ａ）に示すように格納され、多のメモリモジュール１４－ｊにおいて、空間ＩＤ「０１０」に属するデータ群が、図１０（ｂ）に示すように格納されている場合を考える。たとえば、メモリモジュール１４－ｉにおいては、論理アドレス「０」から「５９」までのデータが、そのＲＡＭコアの物理アドレス「１００」から記憶されていることがわかる。この場合に、みかけの配列は、図１０（ｃ）に示すようなものとなる。

【００３８】このように複数のメモリモジュールに、ある配列が格納されている場合に、特定の要素を削除する際の処理につき以下に述べる。ＣＰＵ１２から、各メモリモジュール１４－１、１４－２、…に、制御信号ライン２５を介して、空間ＩＤ「０１０」の要素「５０～５９」を削除するというインストラクションが発せられた場合を考える。図１１および図１３は、ある空間ＩＤ中の所定の範囲の要素を削除するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【００３９】各メモリモジュールのＭＰＵ３６は、制御信号ライン２５を介して与えられたインストラクションを受理して、その内容を解釈し（ステップ１１０１）、インストラクション中の「空間ＩＤ」を調べ（ステップ１１０２）、自己のＲＡＭコア３４が保持するデータの空間ＩＤに関連しているか否かを判断する（ステップ１１０３）。ステップ１１０３にてノー（No）と判断された場合には、処理を終了し、その一方、イエス（Yes）と判断された場合には、ＭＰＵ３６は、空間ＩＤ管理テーブルを参照して、当該空間ＩＤに関するデータ群が書き込み可能な状態になっているか、或いは、削除要求のあった範囲のサイズが、全サイズよりも小さいか否かなどを判断する（ステップ１１０４）。チェックによって異常があると判断された場合（ステップ１１０５でイエス(Yes)）には、ＭＰＵ３６は、制御信号ライン２５を介してエラーが生じたことを通知する。その一方、異常がない場合には、ＭＰＵ３６は、インストラクションにより削除を要求された範囲と、自己のＲＡＭコア３４にて保持する要素の範囲とを比較し（ステップ１１０７）、その比較結果によって（ステップ１１０８）、種々の処理を実行する。

【００４０】まず、削除要求のあった範囲が、自己の保持する要素の範囲よりも後ろである場合（図１１の「Ａ」および図１２（ａ）参照）には、ＭＰＵ３６は何ら処理を実行しない（ステップ１１０９参照）。削除要求のあった範囲が、自己の保持する要素の後方に重なって位置している場合（図１１の「Ｂ」および図１２（ｂ）参照）には、ＭＰＵ３６は、割り付け領域サイズを更新する（ステップ１１１０）。すなわち、削除要求範囲の先頭（矢印１２０１参照）から、自己のＲＡＭコア３４にて保持する要素の範囲の末尾（矢印１２０２参照）までがガーベージとなるように、割り付け領域サイズが変更される。

【００４１】その一方、削除要求のあった範囲が、自己の保持する要素の範囲よりも前方である場合（図１１の「Ｃ」および図１２（ｃ）参照）には、ＭＰＵ３６は、論理開始アドレスを、削除要求のあったサイズ分だけ減じるように、論理開始アドレスを更新する（ステップ１１１１）。さらに、削除要求のあった範囲が、自己の保持する要素の範囲よりも前方で、かつ、一部だけ重なる場合（図１１の「Ｄ」および図１２（ｄ）参照）には、ＭＰＵ３６は、論理開始アドレスを、削除要求のあった範囲の先頭の値に変更するとともに、物理開始アドレスを、削除要求のあった範囲の末尾の値「＋１」に対応する物理アドレスに変更する（ステップ１１１２）。次いで、ＭＰＵ３６は、割り付け領域サイズを更新する（ステップ１１１３）。

【００４２】また、削除要求のあった範囲が、自己の保持する要素の範囲を包含する場合（図１１の「Ｅ」および図１２（ｅ）参照）には、ＭＰＵ３６は、当該空間ＩＤに関する種々のデータを、空間ＩＤ管理テーブルから削除する（図１３のステップ１１１４）。最後に、削除要求のあった範囲が、自己の保持する要素の範囲に包含される場合（図１１の「Ｆ」および図１２（ｆ）参照）には、空間ＩＤ管理テーブルを二つに分割して、削除範囲より前方に関する種々のデータと、削除範囲より後方に関する種々のデータに関するものを生成する（ステップ１１１５）。或いは、ＭＰＵ３６は、自己のＲＡＭ３４に関して、ガベージコレクションを時刻しても良い。

【００４３】このようにして、ＣＰＵ１２からの単一命令（ある空間ＩＤの削除命令）に応答して、各メモリモジュール１４が動作して、所定のメモリモジュールにて必要な処理が並列的に実行される。次に、ある空間ＩＤを有する配列の末尾に、ある要素を追加する場合につき簡単に説明する。図１４は、ある空間ＩＤの配列の末尾に要素を追加するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。図１４のステップ１４０１～ステップ１

４０６は、図１１のステップ１１０１～ステップ１１０６に対応する。次いで、各メモリモジュール１４のＭＰＵ３６は、追加すべき要素を、自己のＲＡＭコア３４に記憶すべきか否かを判断する（ステップ１４０７）。これは、ＭＰＵ３６が、自己の空間ＩＤ管理テーブルを参照することにより実現できる。ステップ１４０７にてイエス(Yes)と判断された場合には、空間ＩＤ管理テーブル中の必要な値を更新し（たとえば、割り付け領域サイズを、追加する要素数に応じて変更する）、次いで、ＲＡＭセル中の所定の領域に、追加すべき要素を書き込む（ステップ１４０９）。或いは、空間ＩＤ管理テーブルの種々の値を生成して、対応するＲＡＭセル中の領域に、追加すべき要素が書き込まれても良い。

【００４４】次いで、ＭＰＵ３６は、空間ＩＤ管理テーブル中の当該空間ＩＤに関連する「全サイズ」の値を更新する（ステップ１４１０）。ステップ１４０７においてノー(No)と判断された場合にも、空間ＩＤ管理テーブル中の関連する「全サイズ」の値が更新される。配列中の任意の位置に要素を追加する場合にも、削除要求と略同等の処理が、各メモリモジュール１４にて実行される。

【００４５】［多空間メモリのより具体的な動作：配列の結合および分割］次に、図１５（ａ）に示すように、複数の配列を結合したり、或いは、図１５（ｂ）に示すように、単一の配列を複数の配列に分割する場合につき説明を加える。本実施の形態にかかるコンピュータシステム１０においては、ある空間ＩＤ（図１５（ａ）においては空間ＩＤ「１００」）を有する配列、および／または、他の空間ＩＤ（図１５（ｂ）においては空間ＩＤ「１００」）を有する配列が、単一のメモリモジュールのＲＡＭコアに収容されていても良いし、或いは、複数のメモリモジュールのＲＡＭコアに収容されていても良い。図１６は、空間ＩＤ「１０」を有する配列および空間ＩＤ「１１」を有する配列、並びに、これらがメモリモジュール中に収容された状態を示す図である。図１６（ａ）においては、その空間ＩＤが「１０」であり、かつ、各要素のサイズが１０ワードである配列１５０１が示されている。この配列１５０１中の要素は、メモリモジュール１４－１ないし１４－ｘに収容されている。また、図１６（ｂ）においては、その空間ＩＤが「１１」であり、かつ、各要素のサイズが１０ワードである配列１５１０が示されている。この配列１５１０の要素も、メモリモジュール１４－１ないし１４－ｘに収容されている。

【００４６】ＣＰＵ１２が、制御信号ライン２５を介して、「空間ＩＤ「１０」の配列と空間ＩＤ「１１」の配列とを結合する」旨のインストラクションを発すると、各メモリモジュール１４は、これを受理して、自己の保持しているデータの空間ＩＤに関するインストラクションであるか否かを判断する。これらの処理は、図１１のステップ１１０１ないしステップ１１０６と略同様である。次いで、自己の保持しているデータの空間ＩＤが、インストラクションに関連している場合には、メモリモジュールのＭＰＵは、以下の手順にしたがって、配列の結合を実現する。上記図１６に示す場合に、関連する各メモリモジュール１４は、空間ＩＤ「１０」および空間ＩＤ「１１」の双方の要素を保持している場合に、空間ＩＤ「１１」に関する空間ＩＤ管理テーブルの値を更新する。より具体的には、空間ＩＤ「１０」に関する「全サイズ」の値を参照して、その論理開始アドレスを再度算出する（たとえば、図１７の符号１７０１、１７０２参照）。また、関連する各メモリモジュールは、空間ＩＤ管理テーブル中の「全サイズ」の値を、二つの配列をくみ合わせたサイズに対応するものに更新する（たとえば、図１７の符号１７０３参照）。図１７は、このようにして得られた配列１７１０、および、各メモリモジュール１４－１～１４－ｘにおける空間ＩＤ管理テーブル（たとえば、符号１７１１、１７１２参照）を示す図である。

【００４７】図１８は、空間ＩＤ「１０」を有する配列を、空間ＩＤ「１０」を有する配列と、空間ＩＤ「１１」を有する配列に分割する一例を示す図である。図１８（ａ）に示す、空間ＩＤ「１０」を有する配列の分解点を定め、分解点より前方に位置する要素を空間ＩＤ「１０」の配列とするとともに、分解点より後方に位置する要素を空間ＩＤ「１１」の配列とする。

【００４８】この場合にも、ＣＰＵ１２が、制御信号ライン２５を介して、「空間ＩＤ「１０」の配列を、分解点を境にして、空間ＩＤ「１０」の配列と空間ＩＤ「１１」の配列とに分解する」旨のインストラクションを発すると、各メモリモジュール１４は、図１１のステップ１１０１ないしステップ１１０６に略対応する処理を実行し、メモリモジュールのうち、インストラクションに関連するもの（図１８の例では、メモリモジュール１４－１～１４－ｘ）が、所定の処理を実行する。たとえば、ＭＰＵ３６は、分解点より後方に位置する要素を収容している場合に、空間ＩＤ「０１１」に関する空間ＩＤ管理テーブル中の種々の値を作成するとともに、空間ＩＤ「０１０」に関する空間管理ＩＤテーブルのうち、全サイズに関する値を更新する。また、分解点より前方に位置する要素のみを収容している場合にも、メモリモジュールのＭＰＵ３６は、空間ＩＤ「０１０」に関する空間管理ＩＤテーブルのうち、全サイズに関する値を更新する。図１９は、このようにして得られた配列１９０１、１９０２、および、各メモリモジュール１４－１～１４－ｘにおける空間ＩＤ管理テーブル（たとえば、符号１９１１、１９１２および１９１３参照）を示す図である。

【００４９】［多空間メモリのより具体的な動作：パラレルコピー］次に、多空間メモリの下で、場合によって

は組み替え可能バスを利用したパラレルコピーにつき、簡単に説明を加える。たとえば、ＣＰＵ１２からの単一のインストラクションにしたがって、図２０に示すように、一方のメモリモジュール群１４０から、他のメモリモジュール群１４１へのデータのパラレルコピーを実現することができる。パラレルコピーには以下の態様が考えられる。

【００５０】（１）一方のメモリモジュール群１４０には単一のメモリモジュールが含まれ、他方のメモリモジュール群には、複数のメモリモジュールが含まれる場合。

（２）一方のメモリモジュール群１４０に、複数のメモリモジュールが含まれ、他方のメモリモジュール群にも、複数のメモリモジュールが含まれる場合。

【００５１】前者においては、コピー元の要素を収容しているメモリモジュール１４のＭＰＵ３６は、ＣＰＵ１２から制御信号ライン２５を介して与えられたインストラクション（たとえば、ある空間ＩＤを有する配列中の所定の要素を、空間ＩＤ８、９、１０の配列としてコピーせよという指令）を受理して、ＲＡＭコア３４から指定された要素を所定のバス上に出力する。その一方、コピー先となるＭＰＵ３６も、同一のインストラクションの受理に応答して、バスから出力された要素を受理して、これをＲＡＭコア３４の所定の領域に記憶するとともに、自己の空間ＩＤ管理テーブルを更新する。

【００５２】後者においては、複数のバスを利用して、一方のメモリモジュール群１４０中のメモリモジュールからのデータを、それぞれ、他方のメモリモジュール群１４１の対応するメモリモジュールに与えることが可能である。この場合には、ＣＰＵ１２は、スイッチ２８およびスイッチ３０を、所定のメモリモジュール間のデータの授受が可能なように制御すれば良い。

【００５３】［多空間メモリのより具体的な動作：隠れ更新など］本実施の形態にかかる多空間メモリを用いて、添字変換により、入力された添え字を変換して、変換済みの添え字によって、配列を指定し、さらに、配列の要素に値を修飾することができる。ある処理が終了して、コミットすることにより、添字変換や値修飾が不要となったときに、各メモリモジュールのＭＰＵは、当該配列に関する空間ＩＤ管理テーブルを書きかえてリマッピングを実行することにより、瞬時に添字変換を解消することができる。その一方、値修飾自体は、実際のＲＡＭコアに記憶された要素を更新する必要があるため、時間を要する。したがって、各メモリモジュールにおいて、変換済フラグを設け、値修飾が反映された要素が、実際にＲＡＭコアに記憶した後に、当該要素に対応するフラグを「１」にセットされる。このようにすれば、あるプロセスにおいて、変換フラグを参照して、これが「１」である場合には、値修飾を経る必要がなく、その一方、変換フラグが「０」であるバイには、値修飾を経る必要があることを容易に知ることができる。したがって、実質的にコミットを瞬時に実現することができる。さらに、本実施の形態にかかる多空間メモリを用いれば、図２１に示すように、ネスト構造の値修飾についても、変換済フラグを設け、この変換済フラグを参照することにより、値修飾を経る必要の有無を知ることが可能となる。

【００５４】［多空間メモリおよび組み替え可能バスの利用：ソート（その１）］本実施の形態においては、多空間メモリおよび組み替え可能バスを利用することにより、ＣＰＵ１２からの単一のインストラクションに基づき、並列的にソート処理を実行することが可能となる。以下、本実施の形態における並列的なソート処理につき説明を加える。図２３および図２４は、本実施の形態にかかるソート処理の流れを説明するための図である。このソート処理では、大きく分けて、図２３に示す処理（存在数の確定および累計の算出）と、図２４に示す処理（レコード番号の転送）とに分けて考えることができる。

【００５５】この実施の形態にかかるソート処理を実現するために、本実施の形態においては、レコード番号を格納したレコード番号配列、ある項目に関する実際の項目値を格納した値リスト、および、レコード番号配列からの値（レコード番号）を入力として、対応する値リストの格納位置を示すポインタ値を出力するように構成された値リストへのポインタとを利用している。すなわち、レコード番号から、対応する位置の値リストへのポインタ値が参照され、そのポインタ値にしたがって、実際の項目値が指定されるようになっている（図２５参照）。まず、ＣＰＵ１２が、必要なインストラクションを、制御信号ライン２５を介して、各メモリモジュール１４に与えると、各メモリモジュールにて、図１１のステップ１１０１ないしステップ１１０６に略同等の処理が実行される。また、関連するメモリモジュールのうち、レコード番号を格納したメモリモジュールからの通知にしたがって、ＣＰＵ１２は、レコード番号を格納した一連のメモリモジュール（第１のメモリモジュール群２３０１）の出力を、あるバス（「第１のバス」と称する）に接続するように、スイッチ２８、３０を制御する。

【００５６】次いで、値リストへのポインタ配列を格納したメモリモジュールからの通知にしたがって、ＣＰＵ１２は、上記値リストへのポインタ配列を格納した一連のメモリモジュール（第２のメモリモジュール群２３０２）の出力を、あるバス（「第２のバス」と称する）に接続するように、スイッチ２８、３０を制御する。さらに、他の一連のメモリモジュール（第３のメモリモジュール群２３０３）においては、値リストへのポインタと同一サイズ（同じ要素数）の「存在数配列」のための領域が確保され、かつ、各要素が「０」に初期化される。

さらに、第３のメモリモジュール群の入力を、上記第２のバスと接続する。

【００５７】次いで、レコード番号配列の先頭から順に、レコード番号が第１のバスに送出される。これは、第１のメモリモジュール群２３０１において、各メモリモジュールのＭＰＵ３６が、空間ＩＤ管理テーブルを参照して、自己が第１のバスにデータを出力するタイミングを検出して、所定のレコード番号を送出することにより実現される。レコード番号は、第１のバスを介して、第２のメモリモジュール群２３０２を構成するメモリモジュールの各々に与えられる。各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理する値リストへのポインタ配列に関連するレコード番号が入力されたことを検出し、当該入力に対応するポインタ値を第２のバスに出力する。

【００５８】ポインタ値は、第２のバスを介して、第３のメモリーのジュール群を構成するメモリモジュールの各々に与えられる。各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理する値リストのポインタ配列に関連するポインタ値が与えられたことを検出し、存在数配列において、ポインタ値に対応する位置の要素をインクリメントする。この動作を繰り返すことにより、項目値が何度レコード番号により指されているか（ポイントされているか）を知ることができる。上記存在数配列のための一連の処理が終了すると、ソートされたレコード番号を格納する配列を作成するために、一連のメモリモジュールに、一定の領域が確保される。この一連のメモリモジュールを、第４のメモリモジュール群２３０４と称する。ＣＰＵ１２は、先の処理に利用した第３のメモリモジュール群の出力と、第４のメモリモジュール群の入力とを、バス（「第３のバス」と称する）を介して接続するように、スイッチ２８、３０を制御する。

【００５９】このような準備が終了した後に、ソート処理が実行される。より具体的には、レコード番号配列の先頭から、レコード番号が第１のバスを介して、第２のメモリモジュール群を構成するメモリモジュールに与えられる。第２のメモリモジュール群中の所定のメモリモジュールにおいては、ＭＰＵ３６がレコード番号の受理に応答して、ポインタ値を、第２のバスを介して、第３のモジュール群に伝達する。次いで、第３のメモリモジュール群のうち、所定のメモリモジュールにおいて、ＭＰＵ３６が、ポインタ値に基づき、関連する存在数配列を参照して、レコード番号の格納位置を決定する。これにより、レコード番号およびその格納位置が、が当該メモリモジュールから、第３のバスに送出される。したがって、第４のメモリモジュール群の所定のメモリモジュールにおいて、ＭＰＵ３６が、レコード番号を、所定の格納位置に配置する。この処理を繰り返すことにより、第４のメモリモジュール群に、ソートされたレコード番号の配列（図２４の符号２４１０）を作成することができる。

【００６０】たとえば、図２３に示す処理を、パイプライン処理にすることができる。すなわち、第１のバスにおいて、あるレコード番号「ｐ」が伝達されている際に、第２のバスにおいては、レコード番号「ｐ－１」に関するポインタ値「Ｐ（ｐ－１）」が伝達され得る。また、同様に、図２４に示す処理も、パイプライン処理にすることが可能である。この場合にも、第１のバスにおいて、あるレコード番号「ｐ」が伝達されている際に、第２のバスにおいては、レコード番号「ｐ－１」に関するポインタ値「Ｐ（ｐ－１）」が伝達され得る。さらに、同じタイミングで、第３のバスにおいては、レコード番号「ｐ－１」に関する格納位置が伝達され得る。

【００６１】このようなパイプライン処理の処理時間につき、以下のような結果が得られた。まず、図２３の処理に関して、第１のバスないし第４のバスが、それぞれ、１２８ビットであり、それぞれ、１２．８ＧＢ／秒の転送能力があると考え、また、レコード番号やポインタ値が、それぞれ、３２ビット整数であると仮定した。いま、レコード数が１０億個の場合に、上記処理では、４０億バイトの転送が発生するが、パイプライン処理を実行するため、４Ｇ／１２．８Ｇ＝０．３１２５秒にて完了することが分かった。同様に、図２４の処理に関して、同様の転送能力およびデータサイズを仮定すると、レコード数が１０億個の場合に、８０億バイトの転送が発生するが、本実施の形態によれば、パイプライン処理の実行により、８Ｇ／１２．８Ｇ＝０．６２５秒にて処理を完了することができる。

【００６２】［多空間メモリおよび組み替え可能バスの利用：ソート（その２）］次に、他の手法によるソート処理につき簡単に説明を加える。このソート処理においても、まず、レコード番号配列を格納したメモリモジュールからなる第１のメモリモジュール群（図２６の符号２６０１参照）の出力と、第１のバスとが接続され、かつ、値リストへのポインタ配列を格納したメモリモジュールからなる第２のメモリモジュール群２６０２の入力が、第１のバスと接続される。これにより、第１のメモリモジュール群２６０１の出力が、第１のバスを介して、第２のメモリモジュール群２６０２に伝達可能となる。その一方、第２のメモリーモジュール群２６０２と、同一の数の空間ＩＤを有する配列の領域が、第３のメモリモジュール群２６０３に確保されるとともに、第２のメモリモジュール群２６０２の出力と、第３のメモリモジュール群の入力とが、第２のバスを介して接続される。

【００６３】次いで、第１のメモリモジュール群２６０１において、あるレコード番号を収容するメモリモジュールのＭＰＵ３６が、当該レコード番号を、第１のバスに送出すると、第２のメモリモジュー群２６０２の所定

のメモリモジュールにおいて、ＭＰＵ３６がこの受理に応答して、対応するポインタ値から、空間ＩＤを算出し、レコード番号および空間ＩＤを、第２のバスに送出する。

【００６４】第３のメモリモジュール群において、当該空間ＩＤおよびレコード番号に基づき、所定のメモリモジュール３６が起動し、当該空間ＩＤを有する配列の末尾に、与えられたレコード番号を配置する。このような処理を全てのレコード番号について実行した後に、第３のメモリモジュール群において、各メモリモジュールのＭＰＵ３６は、自己の有する配列を結合するための処理を実行する。このような手法によっても、高速なソート処理を実現することができる。

【００６５】［多空間メモリおよび組み替え可能バスの利用：検索（その１）］また、本実施の形態においては、多空間メモリおよび組み替え可能バスを利用することにより、ＣＰＵ１２からの単一のインストラクションに基づき、並列的に検索処理を実行することができる。図２７および図２８は、本実施の形態にかかる検索処理の流れを説明するための図である。この検索処理のために、レコード番号配列、値リストへのポインタ配列、値リストおよび後述する可否フラグ配列などが利用される。したがって、この例でも、図２５のように、レコード番号、ポインタ値、項目値の順に、値が参照されるようになっている。

【００６６】まず、ＣＰＵ１２が、必要なインストラクションを、制御信号ライン２５を介して、各メモリモジュール１４に与えると、各メモリモジュールにて、図１１のステップ１１０１ないしステップ１１０６に略同等の処理が実行される。また、関連するメモリモジュールのうち、値リストを格納したメモリモジュールからの通知にしたがって、ＣＰＵ１２は、値リストを格納した一連のメモリモジュール（第１のメモリモジュール群２７０１）の出力を、あるバス（「第１のバス」と称する）に接続するように、スイッチ２８、３０を制御する。さらに、そのよう素数が値リストのものと同じである可否フラグ配列のための領域が、一連のメモリモジュール（第２のメモリモジュール群２７０２）に確保され、当該第２のメモリモジュール２７０２に属する各メモリモジュールのＭＰＵ３６が、当該領域の要素を「０」に初期化する。

【００６７】次いで、第２のメモリモジュール群２７０２の入力が、第１のバスに接続される。次いで、ＣＰＵ１２から与えられた検索条件にしたがって、第２のメモリモジュール群の各メモリモジュールにおいて、ＭＰＵ３６が、値リスト中の検索条件に合致する項目値の位置を参照して、可否フラグ配列の対応する値を「１」にセットする。たとえば、検索条件が範囲であれば、二分割法などを用いれば良い。また、その他の条件であれば、要素ごとにその可否を判断すれば良い。このような処理が終了した後に、検索が実行される。まず、レコード番号配列を格納した一連のメモリモジュール（第３のメモリモジュール群２７０３）の出力を、第１のバスに接続するとともに、値リストへのポインタ配列を格納した一連のメモリモジュール（第４のメモリモジュール群２７０４）の入力を、第１のバスに接続するよう、ＣＰＵ１２は、スイッチ２８、３０を制御する。また、第４のメモリモジュール群２７０４の出力を、第２のメモリモジュール群２７０２の入力とを、第２のバスと接続するように、ＣＰＵ１２は、スイッチ２８、３０を制御する。

【００６８】さらに、レコード番号の要素数と同じ要素数を有する配列のための領域が、一連のメモリモジュール（第５のメモリモジュール２７０５）に確保され、ＣＰＵ１２は、その入力と、第２のメモリモジュール群２７０２の出力とが、第３のバスを介して接続されるように、スイッチ２８、３０を制御する。このような処理の後に、レコード番号配列の先頭から順に、レコード番号が第１のバスに送出される。これは、第３のメモリモジュール群２７０３において、各メモリモジュールのＭＰＵ３６が、空間ＩＤ管理テーブルを参照して、自己が第１のバスにデータを出力するタイミングを検出して、所定のレコード番号を送出することにより実現される。

【００６９】レコード番号は、第１のバスを介して、第４のメモリモジュール群２７０４を構成するメモリモジュールの各々に与えられる。各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理する値リストへのポインタ配列に関連するレコード番号が入力されたことを検出し、受理したレコード番号および当該入力に対応するポインタ値を第２のバスに出力する。ポインタ値は、レコード番号とともに、第２のバスを介して、第３のメモリーのジュール群を構成するメモリモジュールの各々に与えられる。各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理する可否フラグ配列の位置と同じ位置を示すポインタ値が与えられたことを検出し、当該ポインタ値が示す可否フラグが、「０」であるか「１」であるかを判断する。次いで、可否フラグが「１」の場合には、関連するレコード番号が、第３のバスを介して、第５のメモリモジュール群２７０５に与えられる。

【００７０】第５のメモリモジュール群２７０５においては、各メモリモジュールのＭＰＵ３６は、自己の空間ＩＤ管理テーブルを参照して、自己が管理するヒット情報格納用配列の位置と同じ位置を示すレコード番号が与えられたことを検出し、その位置の要素を「１」にする。このような処理を所定のレコード番号に関して繰り返し、ヒット情報格納用配列にて「１」である要素を取り出すことにより、検索が完了する。

【００７１】ソート処理と同様に、上記検索処理でも、図２７を参照して説明した処理、および、図２８を参照

して説明した処理を、それぞれ、パイプライン処理にて実現することができる。検索処理におけるパイプライン処理の処理時間につき、以下のような結果が得られた。バスの転送能力、および、各要素のビット数は、ソート処理と同様であると考えた。レコード数が１０億個の場合に、上記検索処理では、８０億バイトの転送が発生するが、パイプライン処理を実行するため、８Ｇ／１２．８Ｇ＝０．６２４秒にて完了することが分かった。

【００７２】さらに、本検索処理を用いれば、ＡＮＤ、ＯＲ或いはＮＯＴなどを組み合わせた複数項目の検索を実現することもできる。より具体的には、各項目につき、ヒット情報格納用配列を作成し、これら配列の要素間での論理演算を行えば良い。たとえば、二つの項目のＡＮＤ或いはＯＲ検索では、ヒット情報格納用配列の要素の転送（１０億バイト）が行われる。したがって、その処理時間は、（１０Ｇ／８）／１２．８Ｇ＝０．０９８秒だけ必要であることが理解できる。

【００７３】なお、さらに高速化を図るために、ＡＮＤ検索の場合には、二つの検索処理を実行するメモリモジュール群を縦列に接続すれば良い。また、第４のメモリモジュール群と第２のメモリモジュール群を、同一の複数のメモリモジュールにて構成できるように、配列を配置すれば、ボトルネックを解消することができ、これにより、略２倍の処理速度を得ることが可能となる。

【００７４】本発明は、以上の実施の形態に限定されることなく、特許請求の範囲に記載された発明の範囲内で、種々の変更が可能であり、それらも本発明の範囲内に包含されるものであることは言うまでもない。たとえば、前記実施の形態においては、本発明を、コンピュータシステムに適用しているがこれに限定されるものではなく、パーソナルコンピュータなどに接続可能なコンピュータボードに適用することもできる。この場合には、図１において、ＣＰＵ１２、メモリユニット１４、バス２４等がボード上に搭載され、これが、本発明における情報処理ユニットを構成する。

【００７５】また、ＣＰＵ１２とメモリモジュール１４との間、および／または、メモリモジュール１４間を接続するバスの組の数は、前記実施の形態に限定されるものではなく、コンピュータシステムを搭載する回路基板の大きさ、各バスのビット数などを考慮して適宜決定することができる。また、前記実施の形態においては、メモリモジュールの入出力とバスとの接続を規定するためのスイッチ２８と、ＣＰＵとメモリモジュールとの間、メモリモジュール間で、バスの切断することができるスイッチ３０とを設けている。スイッチ３０を設けることにより、たとえば、あるバス（図１のバス２４－４参照）を、ＣＰＵモジュール１２とメモリモジュール１４－１とのデータ授受のために利用するとともに、同時に、メモリモジュール１４－２とメモリモジュール１４－３との間のデータ授受のために利用することができる（この場合に、スイッチ３０－５をオフにすれば良い）。したがって、より有効にバスを利用することが可能となっている。しかしながら、バスの組を数を十分に大きくできる場合、或いは、メモリモジュールの数が比較的少ない場合には、スイッチ３０を必ずしも設けなくて良い。

【００７６】また、本明細書において、制御信号ライン２５を介して、ＣＰＵ１２からのインストラクションが与えられることを記載したが、制御信号ライン２５を介して、インストラクションのほか、クロックなど、各メモリモジュールが同期して作動するための種々の制御信号が与えられ、かつ、各メモリモジュールからＣＰＵ１２への所定の信号（たとえば、エラー信号や、データ受理を示す信号）が与えられていることは言うまでもない。さらに、本明細書において、一つの手段の機能が、二つ以上の物理的手段により実現されても、若しくは、二つ以上の手段の機能が、一つの物理的手段により実現されてもよい。

【００７７】

【発明の効果】本発明によれば、分散メモリー型において、単一命令により種々のメモリーに記憶された配列中の要素を入出力し、著しく高速な並列処理を実現可能なコンピュータアーキテクチャを提供することが可能となる。

【図面の簡単な説明】

【図１】　図１は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイヤグラムである。

【図２】　図２は、本実施の形態にかかるメモリモジュールの概略を示すブロックダイヤグラムである。

【図３】　図３は、単一メモリ空間における一連のデータの配置を示す図である。

【図４】　図４は、本発明に係る多空間メモリにおける一連のデータの配置を示す図である。

【図５】　図５は、本実施の形態におけるアドレスリマッピングを説明するための図である。

【図６】　図６は、本実施の形態における値修飾を説明するための図である。

【図７】　図７は、本実施の形態にかかるメモリモジュール間のパイプライン処理の概略を示す図である。

【図８】　図８は、本実施の形態にかかる多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。

【図９】　図９は、多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。

【図１０】　図１０は、多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。

【図１１】　図１１は、ある空間ＩＤ中の所定の範囲の要素を削除するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャ

ートである。

【図１２】　図１２は、削除される要素と、メモリモジュールにて保持している要素の配置との関係を示す図である。

【図１３】　図１３は、ある空間ＩＤ中の所定の範囲の要素を削除するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【図１４】　図１４は、ある空間ＩＤの配列の末尾に要素を追加するというインストラクションを受理した各メモリモジュールにて実行される処理を示すフローチャートである。

【図１５】　図１５は、本実施の形態にかかる配列の結合および配列の分割を説明するための図である。

【図１６】　図１６は、本実施の形態において、空間ＩＤ「１０」を有する配列および空間ＩＤ「１１」を有する配列、並びに、これらがメモリモジュール中に収容された状態を示す図である

【図１７】　図１７は、本実施の形態において、配列の結合により得られた配列、および、各メモリモジュールにおける空間ＩＤ管理テーブルを示す図である。

【図１８】　図１８は、本実施の形態において、空間ＩＤ「１０」を有する配列を、空間ＩＤ「１０」を有する配列と、空間ＩＤ「１１」を有する配列に分割する一例を示す図である。

【図１９】　図１９は、本実施の形態において、配列の分割により得られた配列、および、各メモリモジュールにおける空間ＩＤ管理テーブルを示す図である。

【図２０】　図２０は、本実施の形態かかる、一方のメモリモジュール群から、他のメモリモジュール群へのデータのパラレルコピーを示す図である。

【図２１】　図２１は、本実施の形態にかかる変換済み＊フラグの利用を説明するための図である。

【図２２】　図２２は、本実施の形態にかかる変換済みフラグの利用を説明するための図である。

【図２３】　図２３は、本実施の形態にかかるソート処理の流れを説明するための図である。

【図２４】　図２４は、本実施の形態にかかるソート処理の流れを説明するための図である。

【図２５】　図２５は、本実施の形態において、レコード番号から項目値が特定されるまでのデータの参照手順を示す図である。

【図２６】　図２６は、本実施の形態にかかる他のソート処理の流れを説明するための図である。

【図２７】　図２７は、本実施の形態にかかる検索処理の流れを説明するための図である。

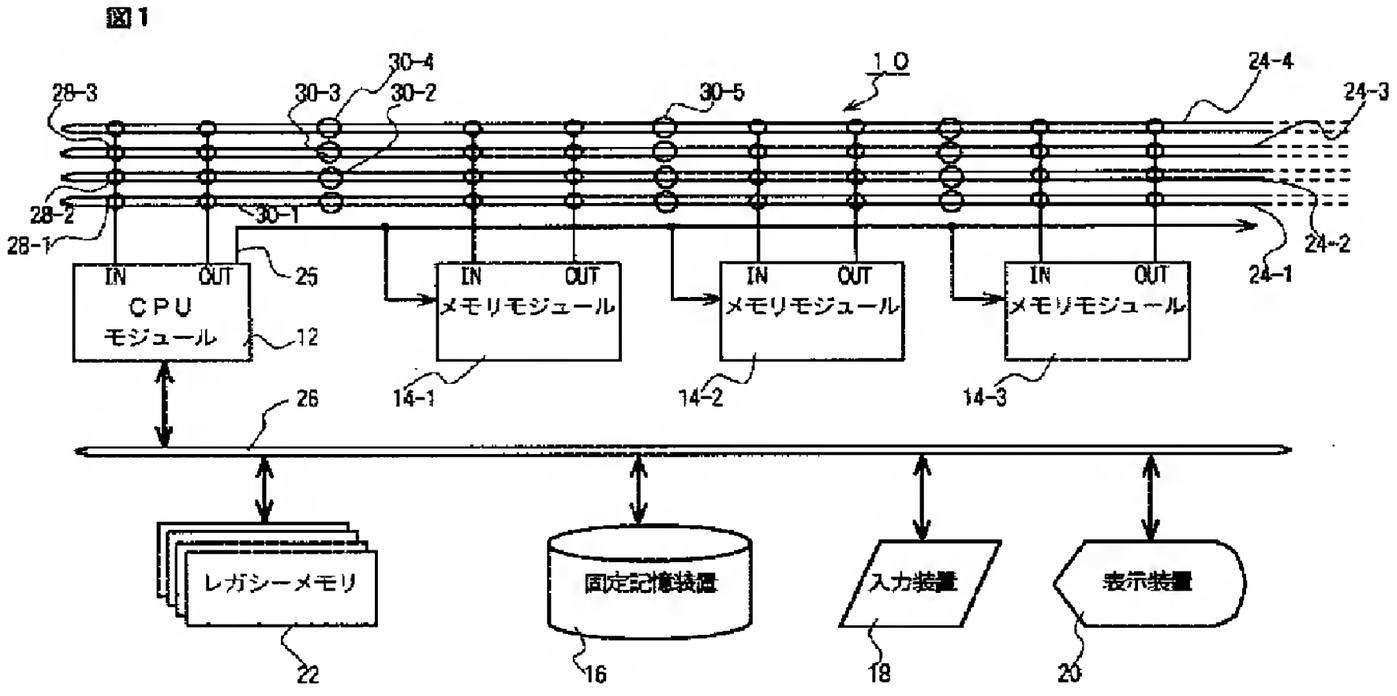
【図２８】　図２８は、本実施の形態にかかる検索処理の流れを説明するための図である。

【符号の説明】

１０	コンピュータシステム
１２	ＣＰＵモジュール
１４	メモリモジュール
１６	固定記憶装置
１８	入力装置
２０	表示装置
２２	レガシーメモリ
２４	バス
２５	制御信号ライン
２６	バス
２８、３０	スイッチ
３２	クロックバッファ
３４	ＲＡＭコア
３６	ＭＰＵ
３８	Ｉ／Ｏ

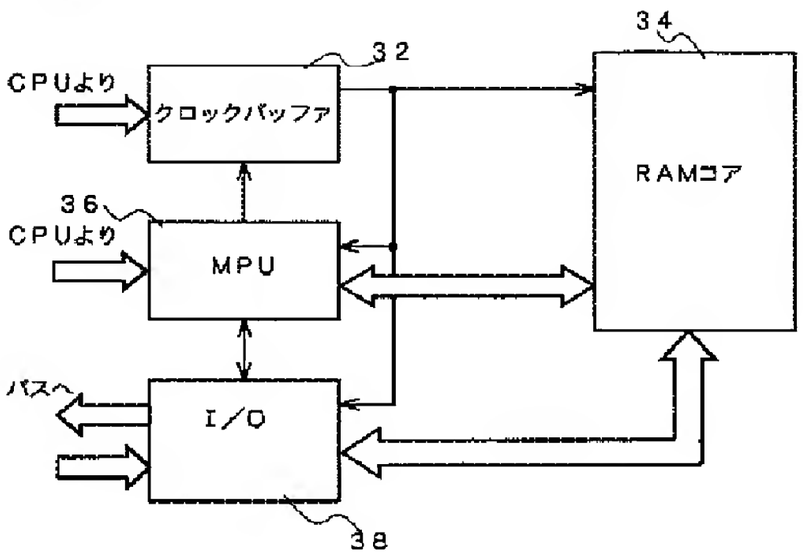
【図１】

図１



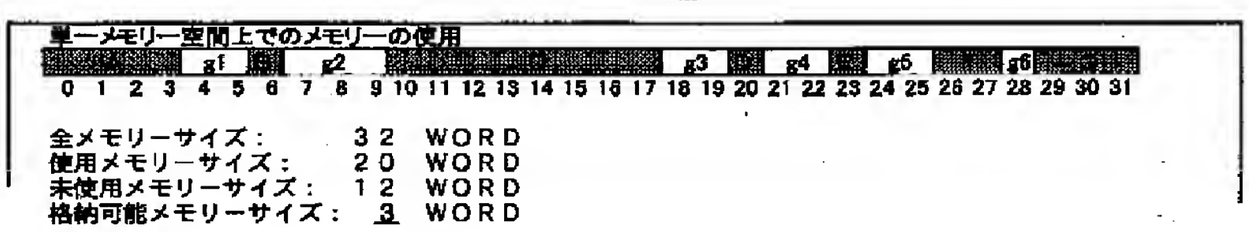
【図２】

図2



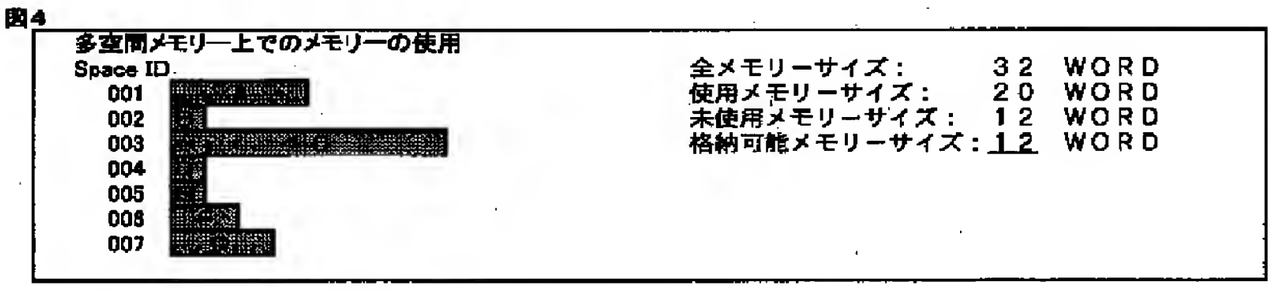
【図３】

図3



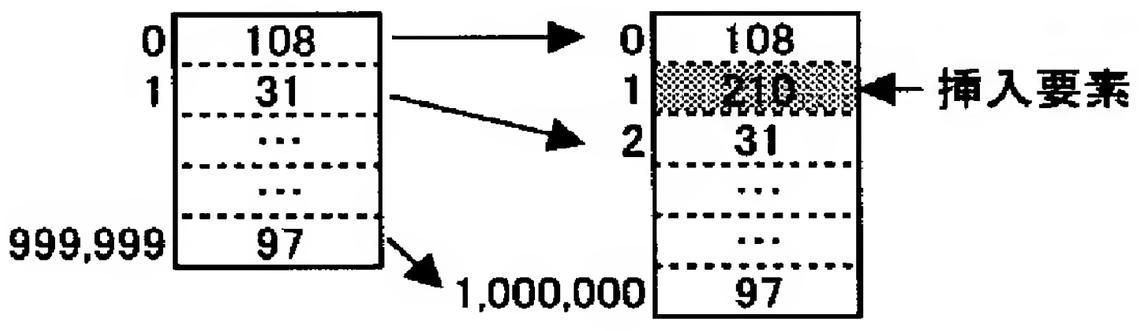
【図４】

図4



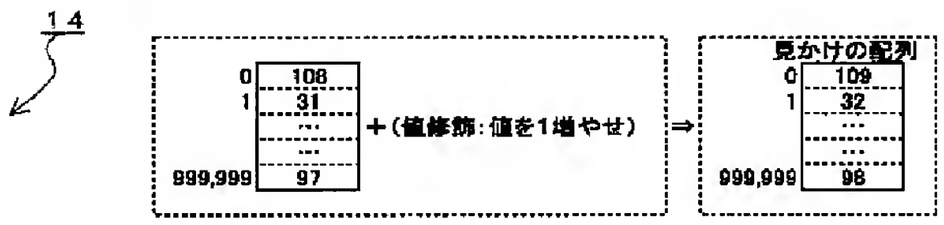
【図５】

図5



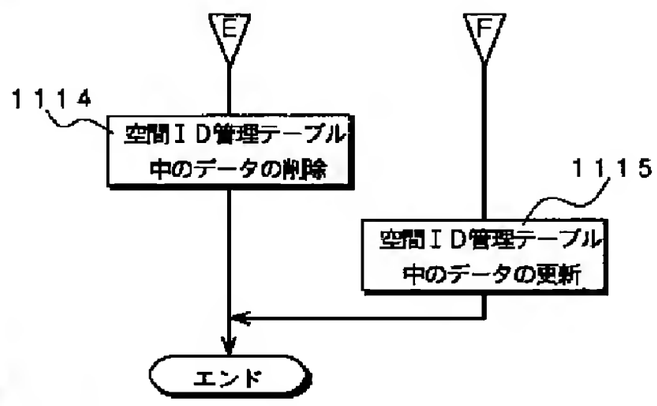
【図６】

図6



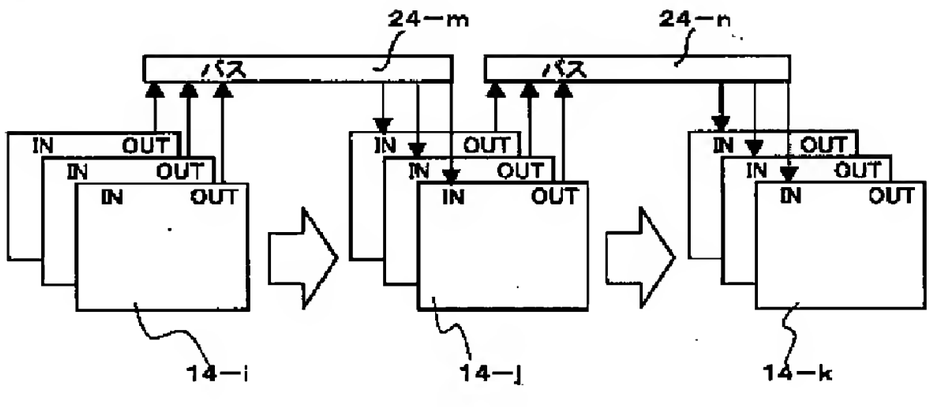
【図１３】

図13



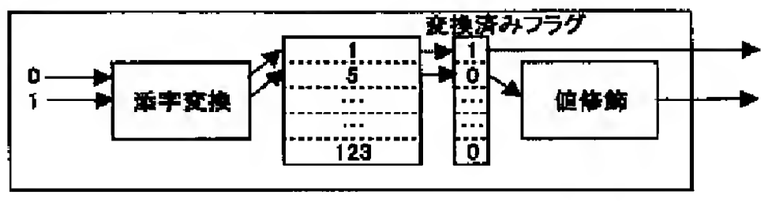
【図７】

図7



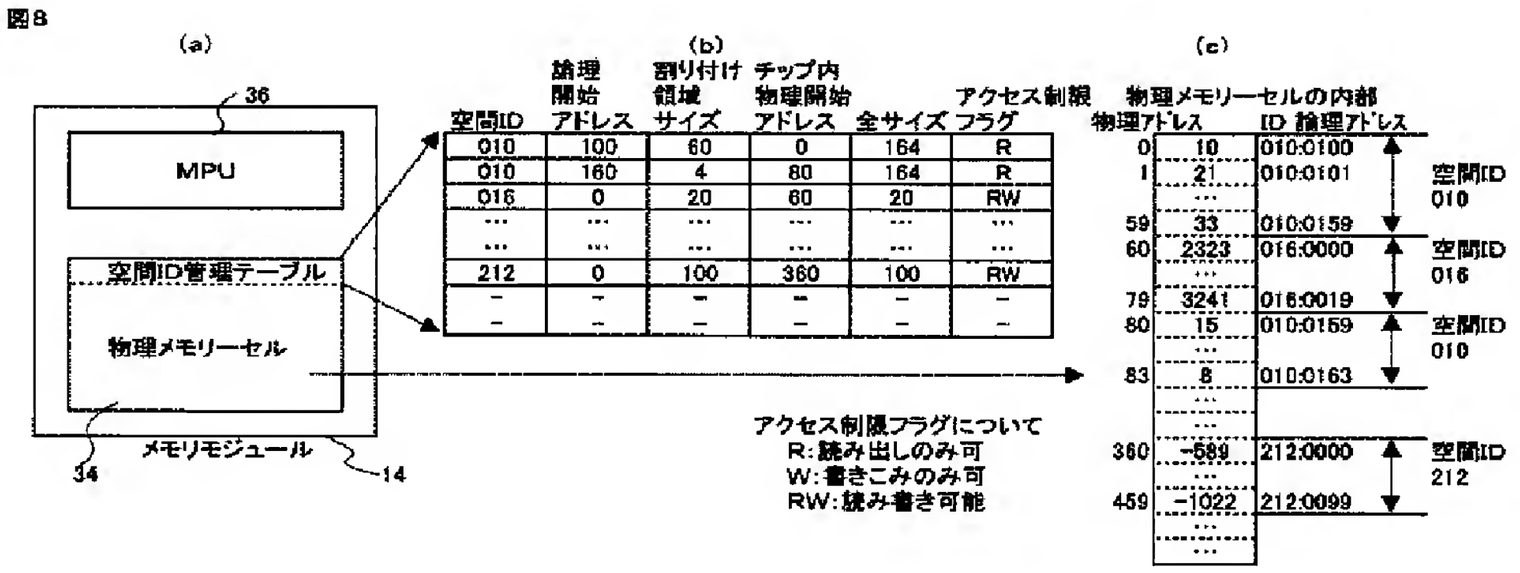
【図２１】

図21



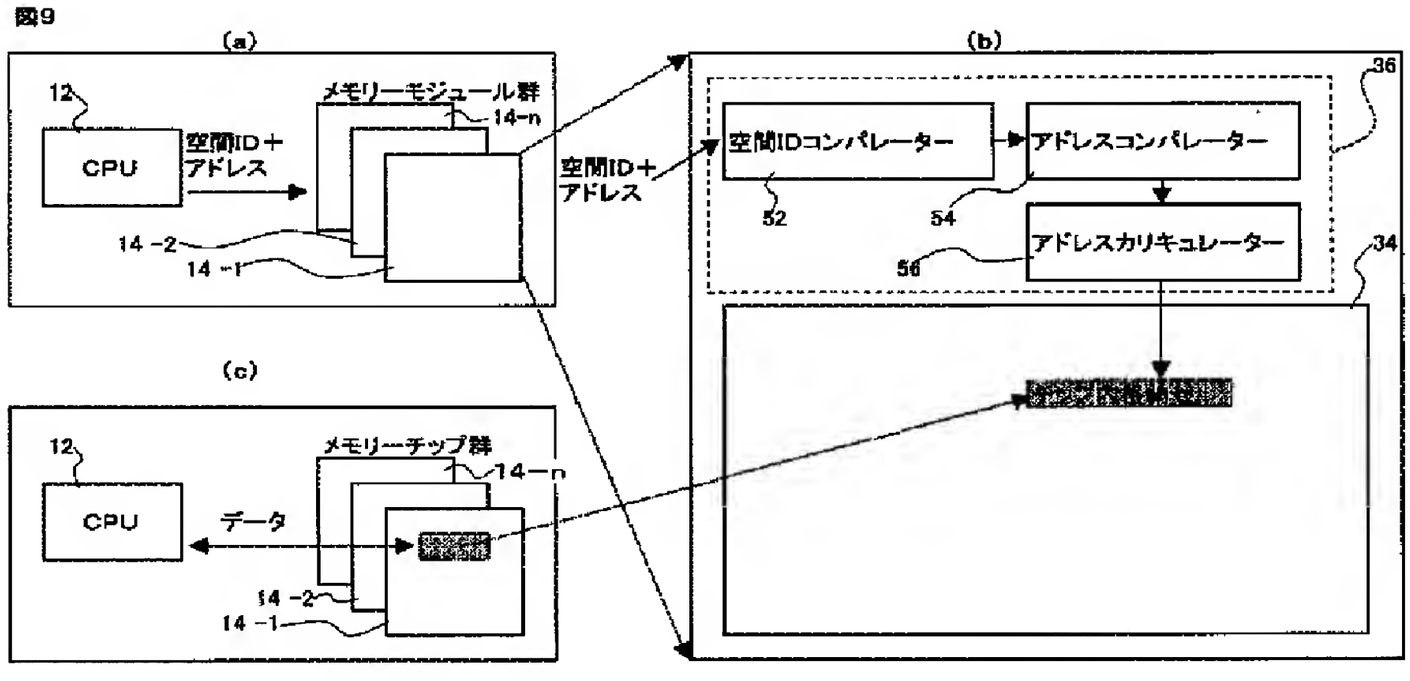
【図８】

図8



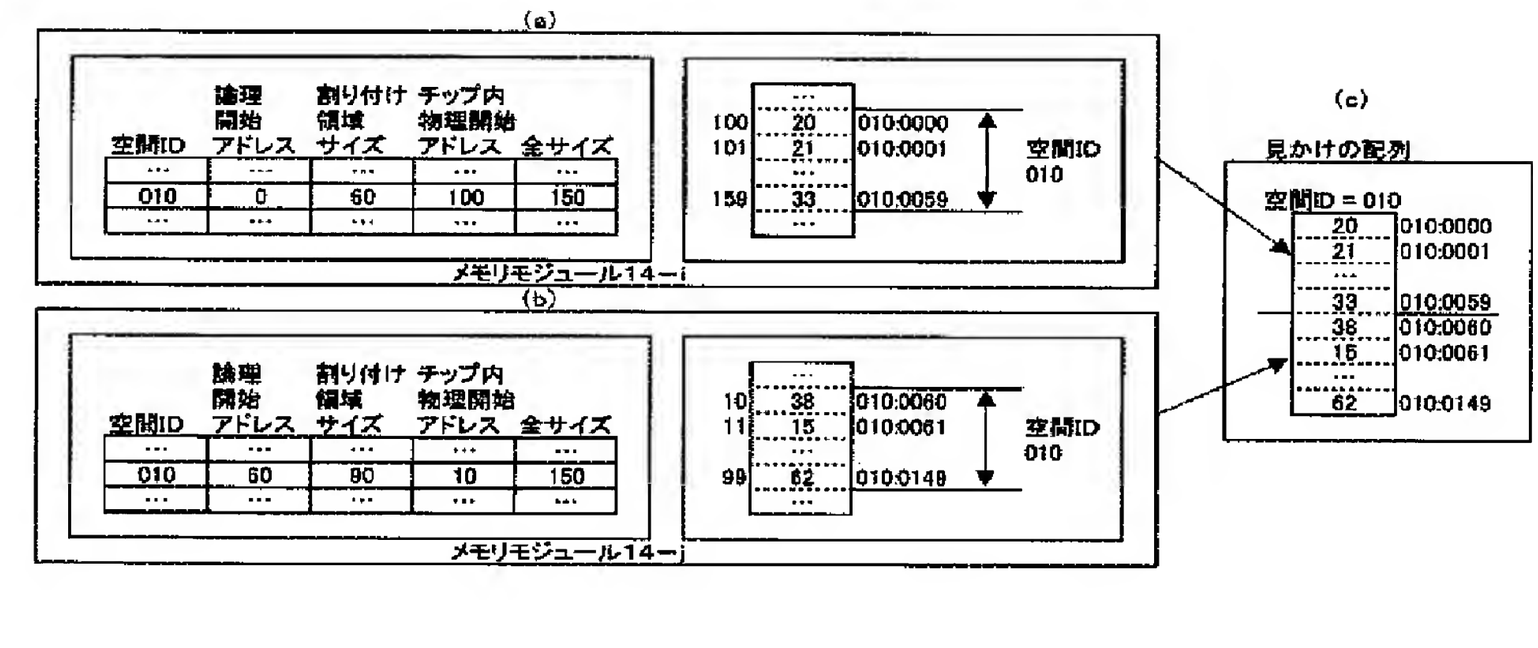
【図９】

図9



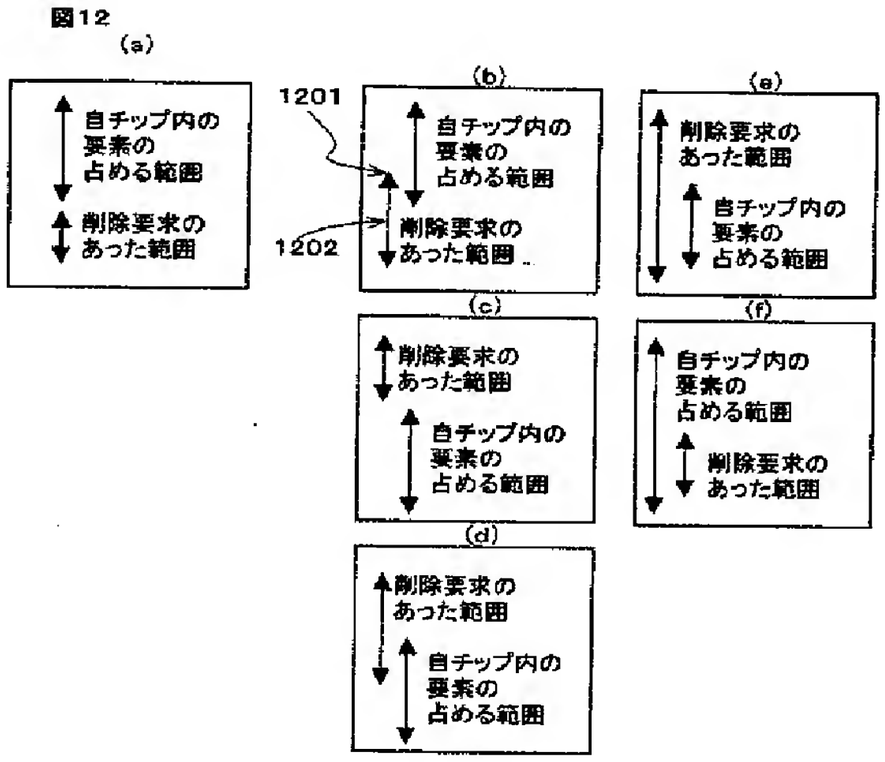
【図１０】

図10



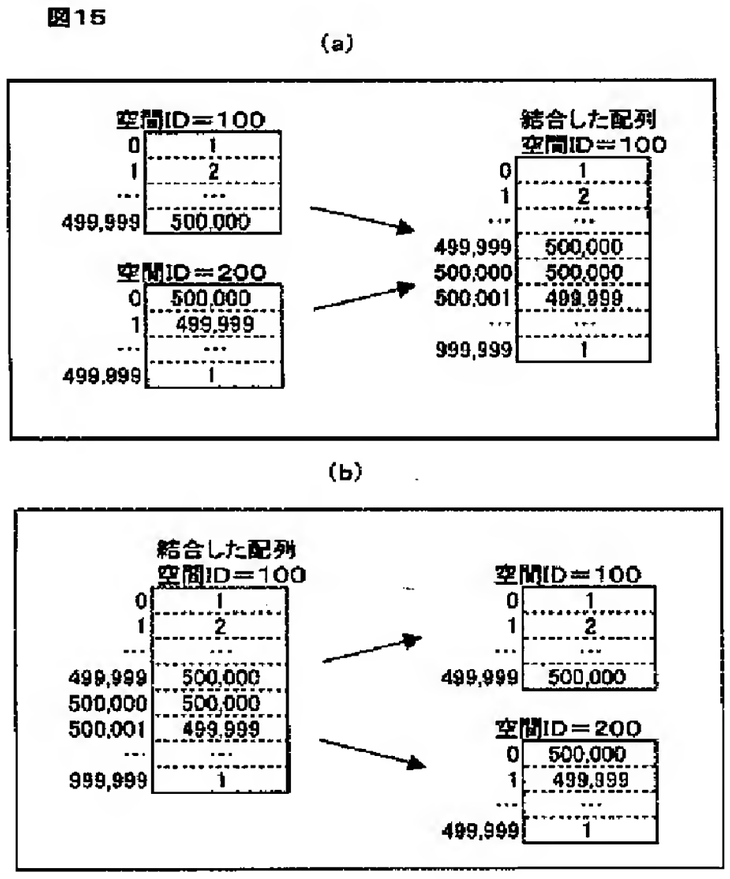
【図１２】

図12



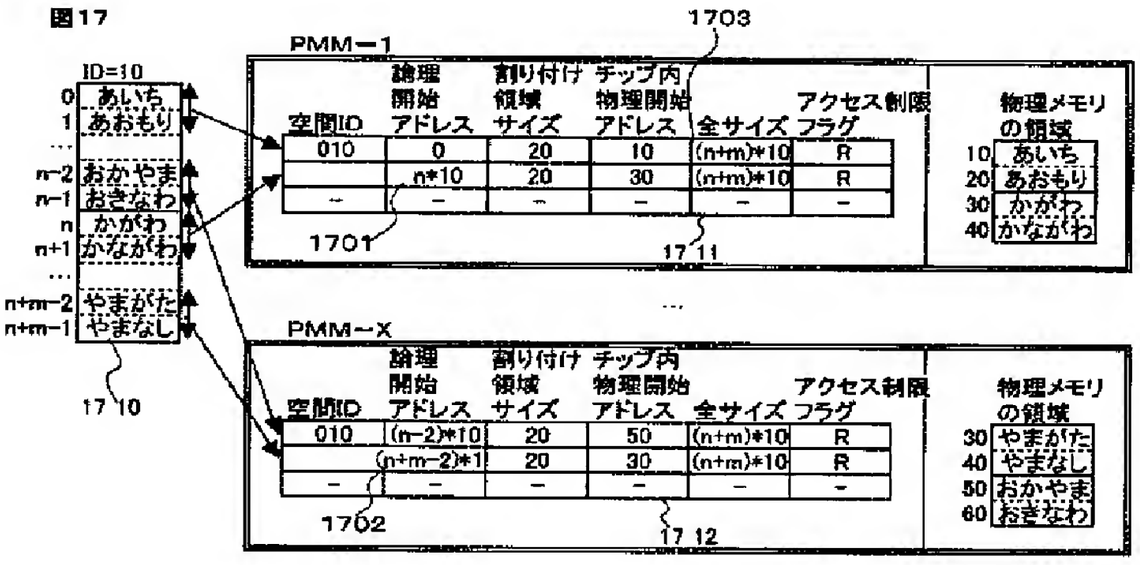
【図１５】

図15



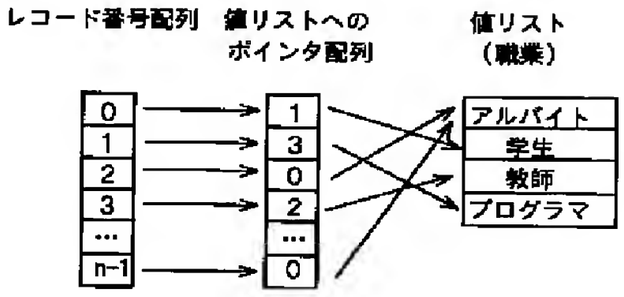
【図１７】

図17



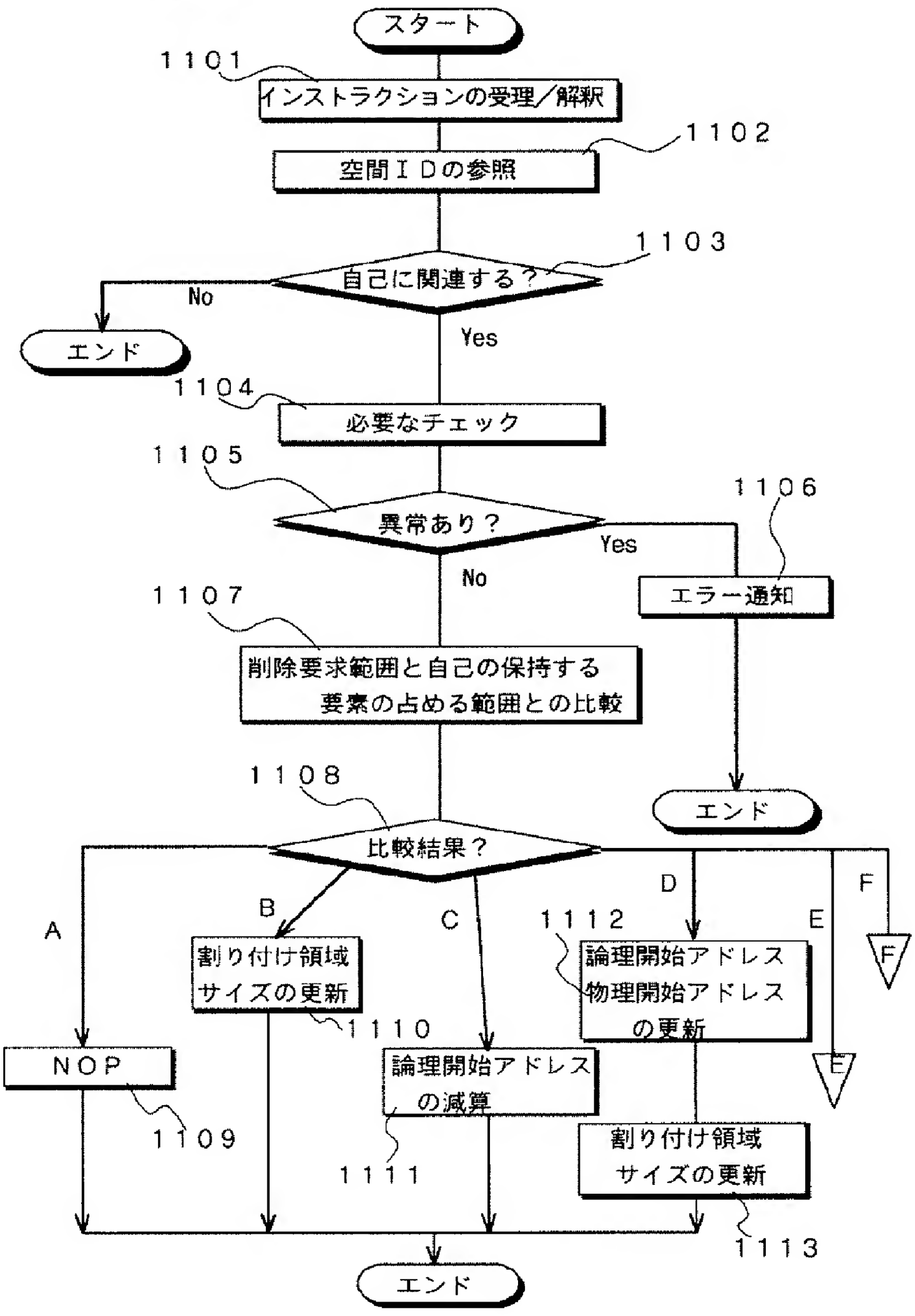
【図２５】

図25



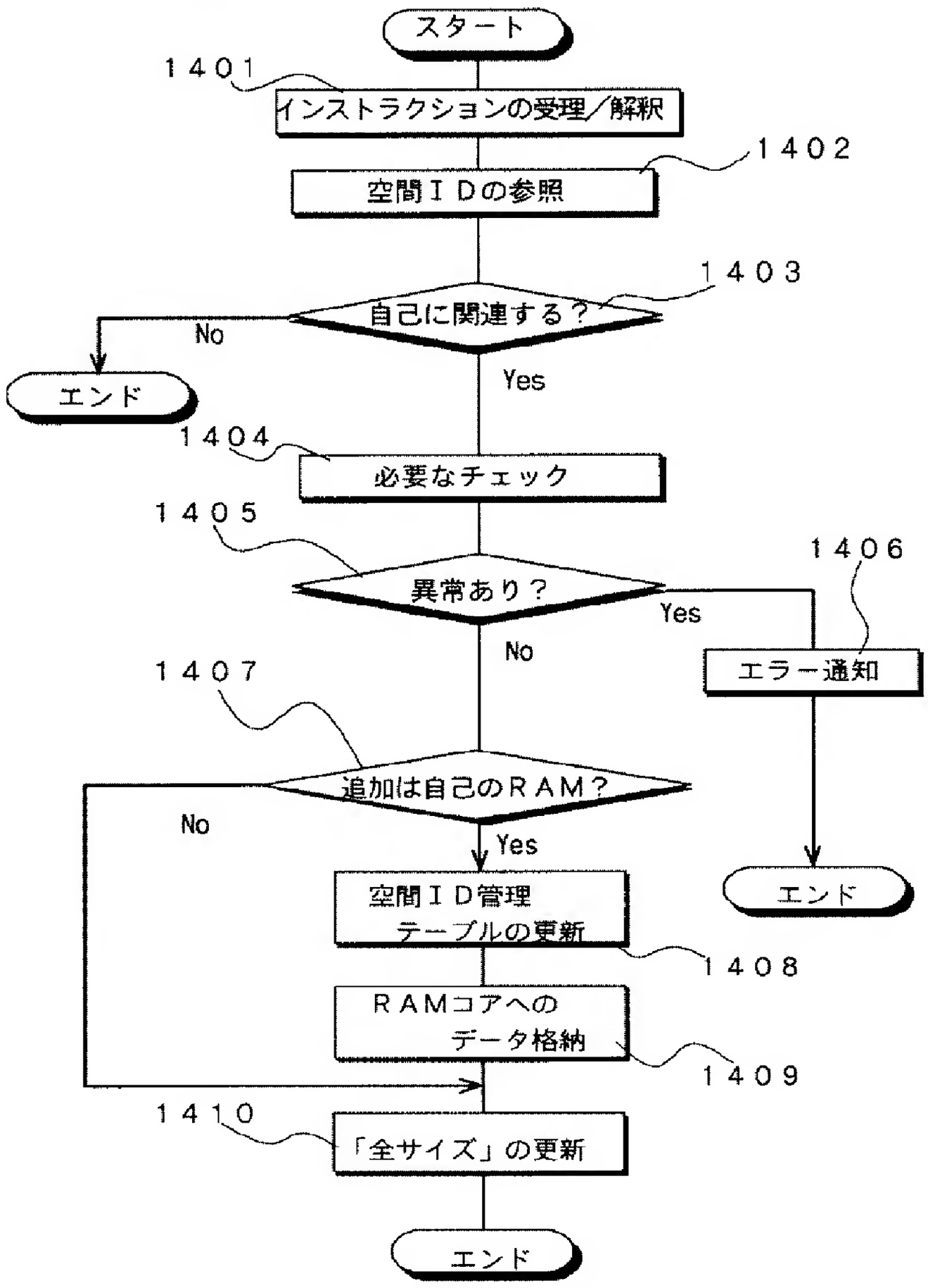
【図１１】

図１１


スタート
1101
インストラクションの受理／解釈
1102
空間ＩＤの参照
1103
自己に関連する？
No
Yes
エンド
1104
必要なチェック
1105
異常あり？
1106
Yes
No
エラー通知
1107
削除要求範囲と自己の保持する
要素の占める範囲との比較
エンド
1108
比較結果？
A
B
C
D
E
F
NOP
1109
割り付け領域
サイズの更新
1110
論理開始アドレス
の減算
1111
1112
論理開始アドレス
物理開始アドレス
の更新
割り付け領域
サイズの更新
1113
F
E
エンド

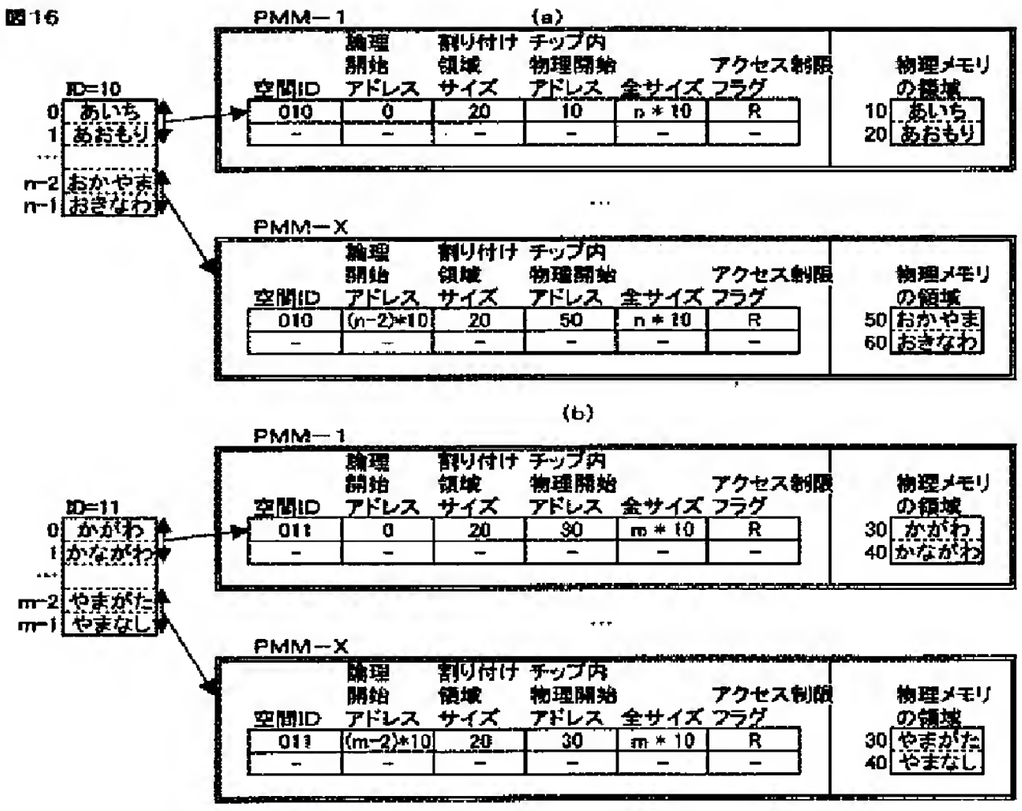
【図１４】

図１４


スタート
1401
インストラクションの受理／解釈
1402
空間ＩＤの参照
1403
自己に関連する？
No
エンド
Yes
1404
必要なチェック
1405
異常あり？
1406
Yes
エラー通知
エンド
No
1407
追加は自己のＲＡＭ？
No
Yes
空間ＩＤ管理
テーブルの更新
1408
ＲＡＭコアへの
データ格納
1409
1410
「全サイズ」の更新
エンド

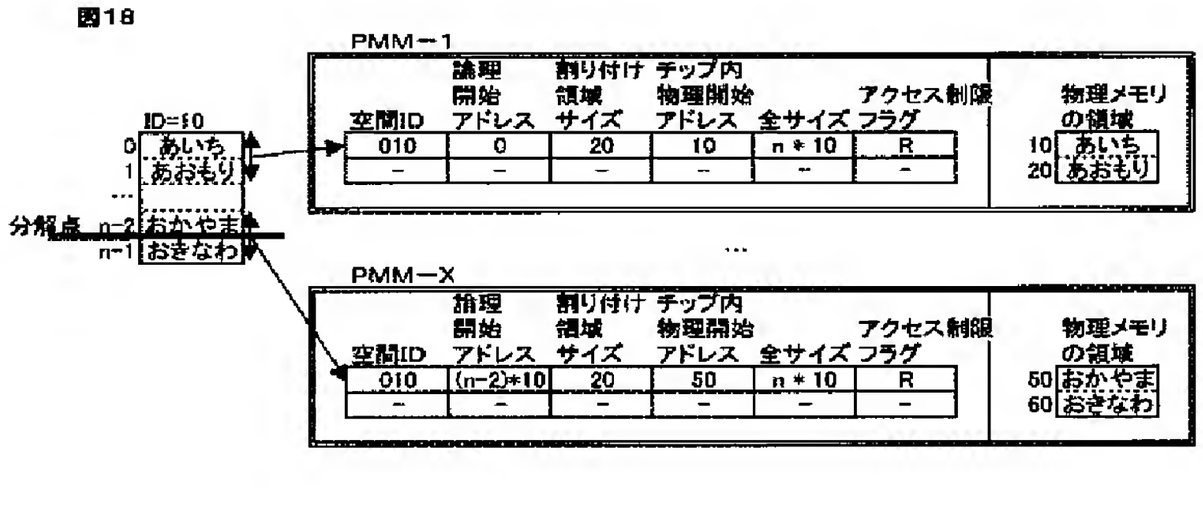
【図１６】

図16



【図１７】

図17

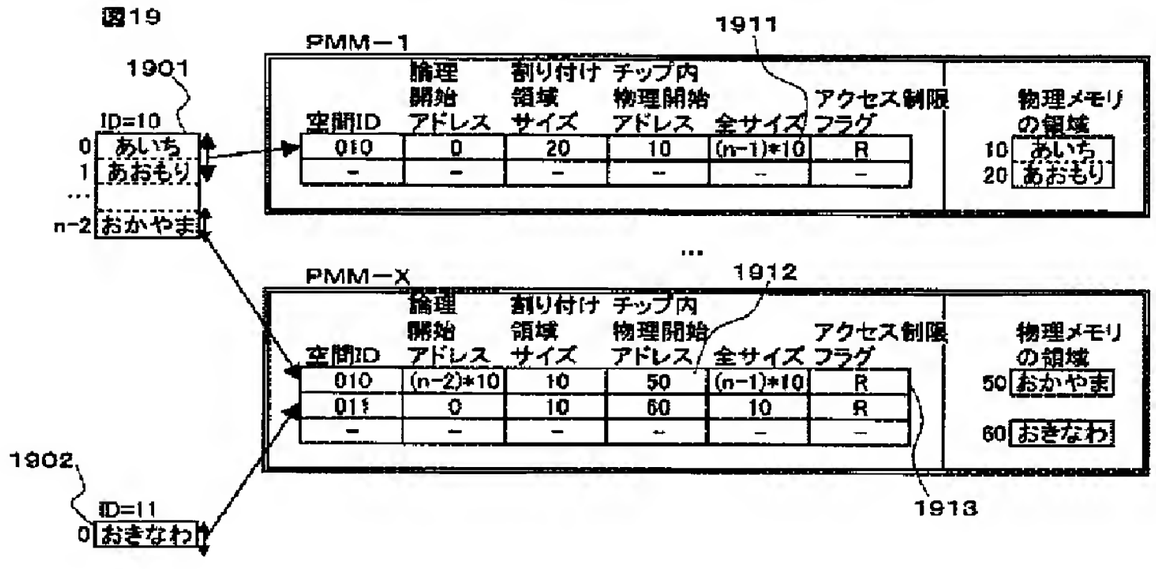


【図１８】

図18

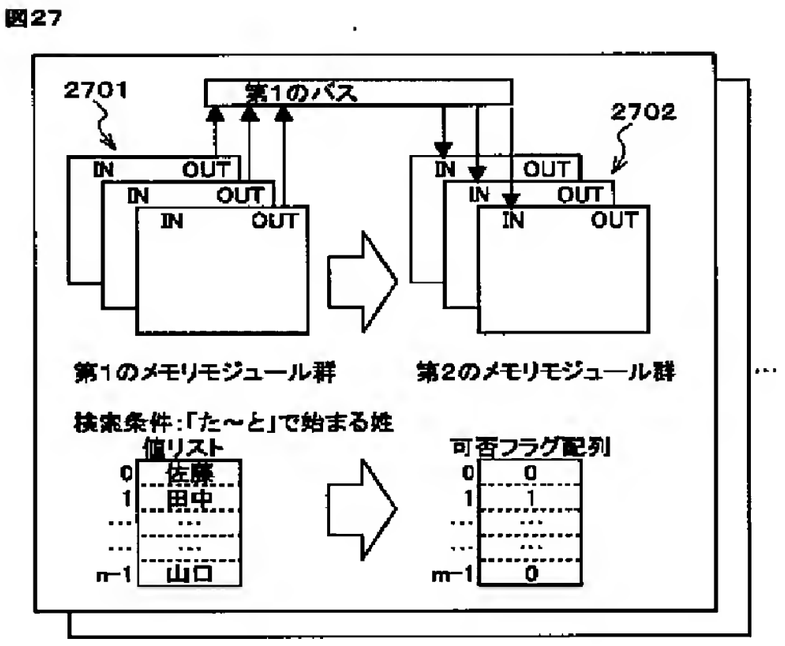
【図１９】

図19



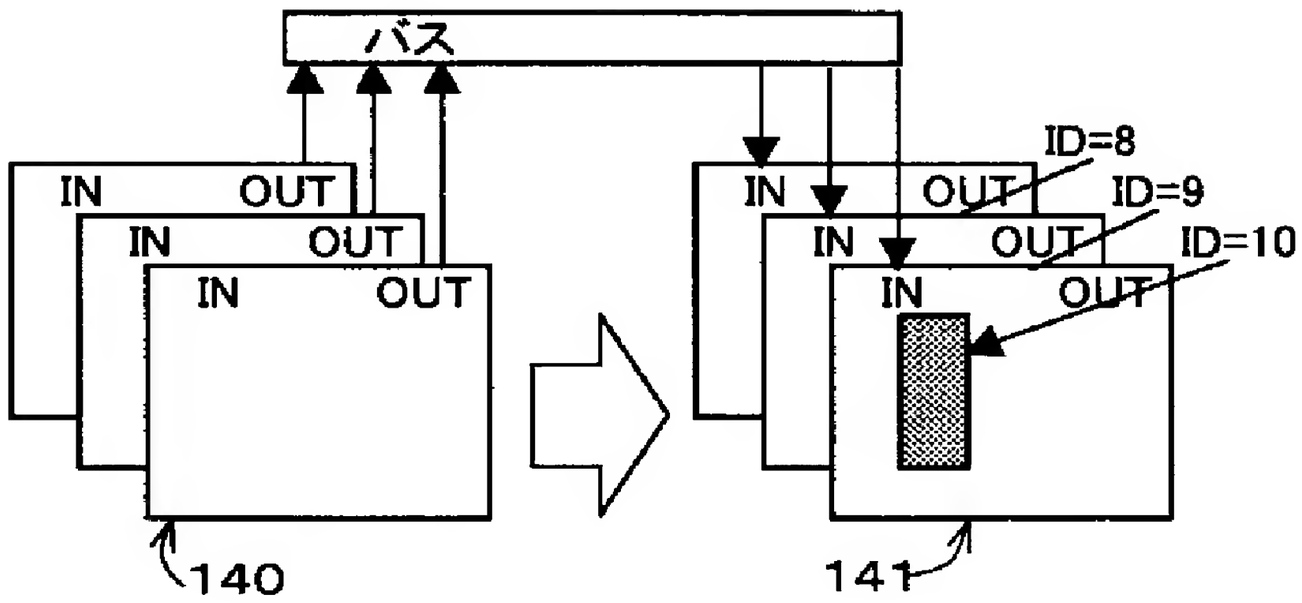
【図２７】

図27



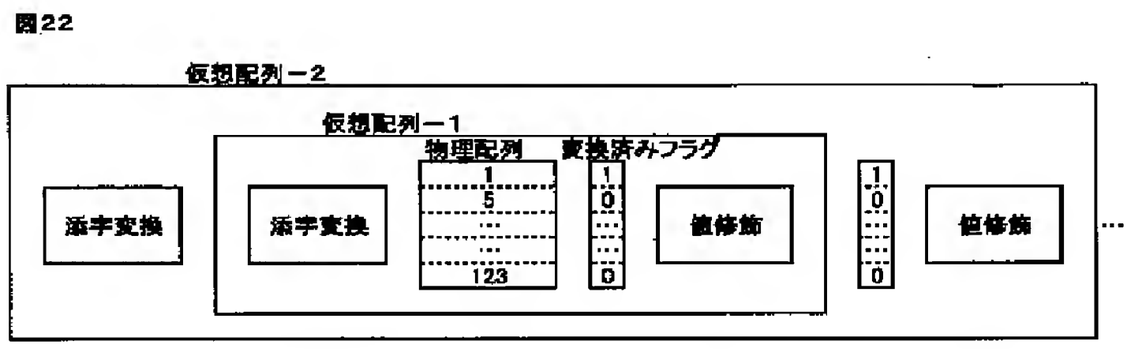
【図２０】

図20



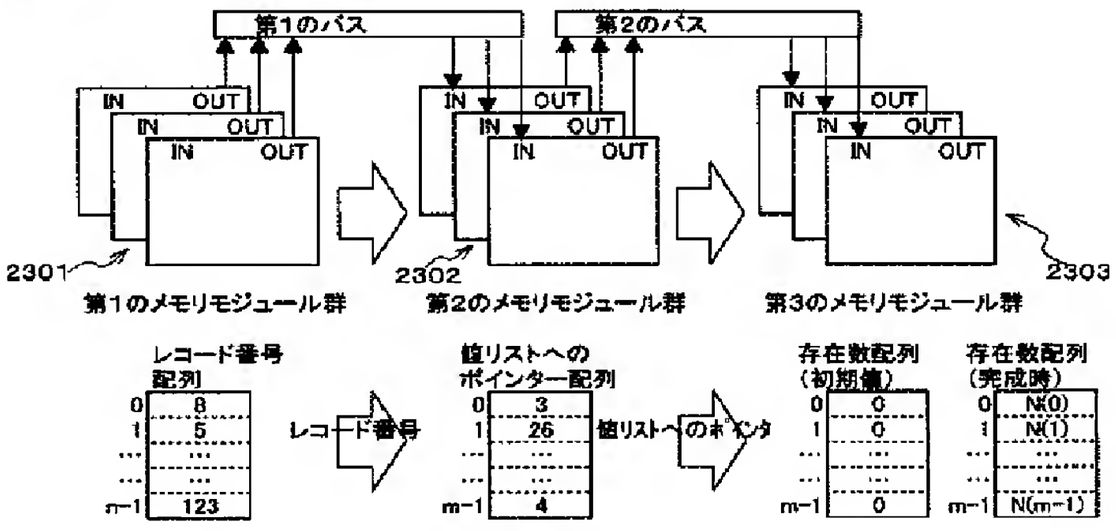
【図２２】

図22



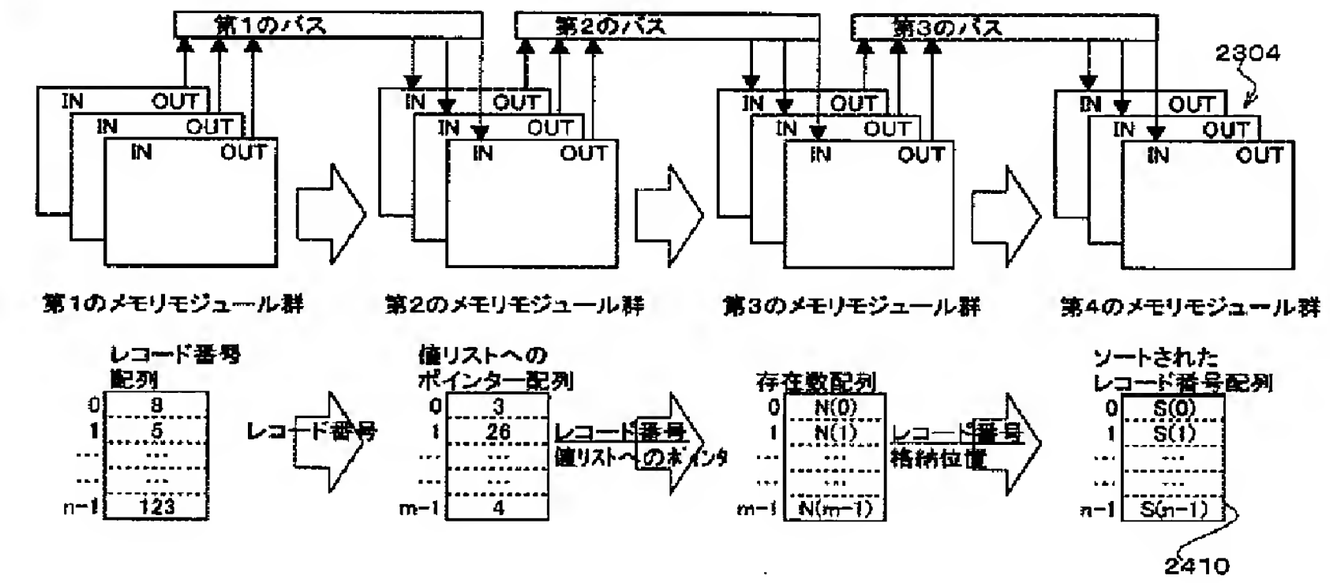
【図２３】

図23



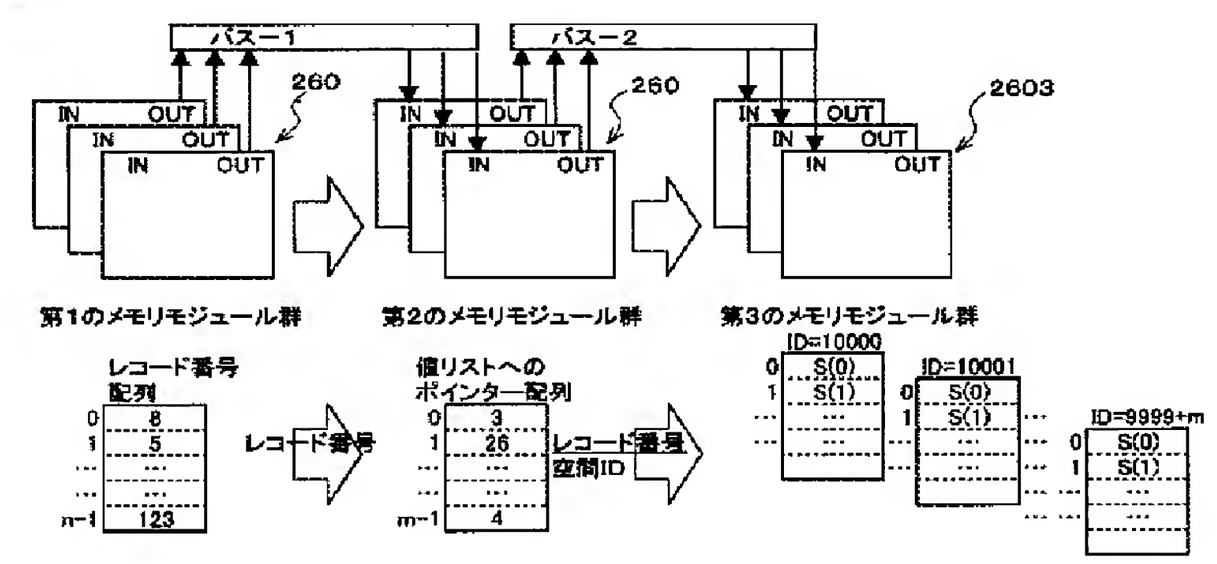
【図２４】

図24



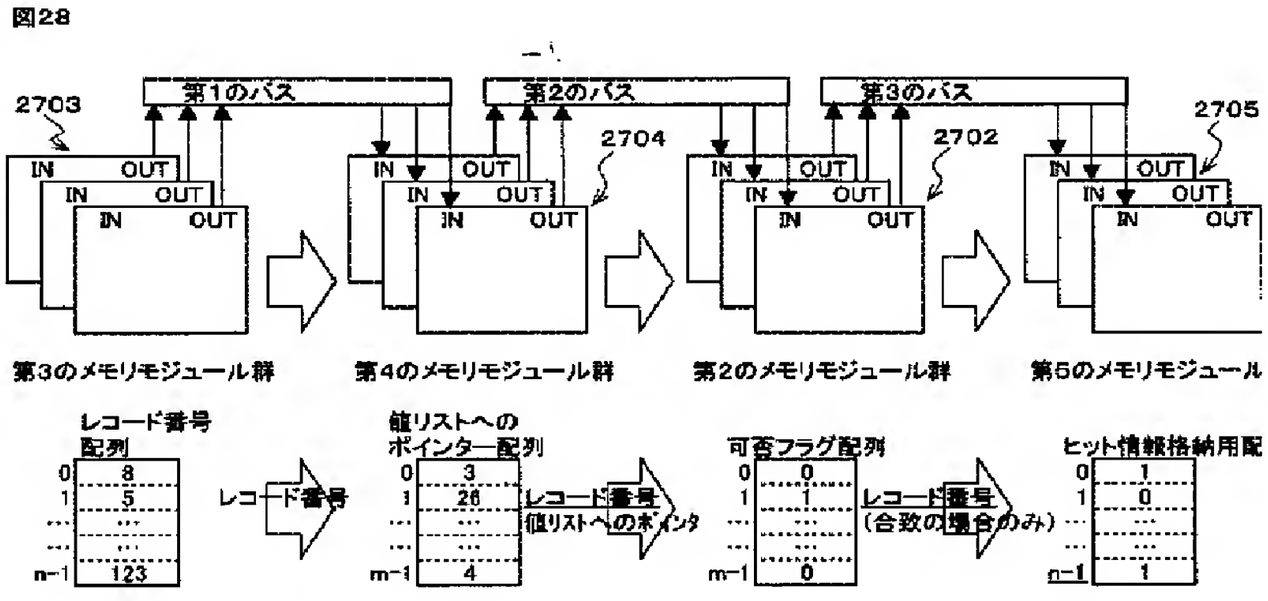
【図２６】

図26



【図２８】

図28



フロントページの続き

(51)Int.Cl.[7]		識別記号	F I		テーマコード（参考）
Ｇ０６Ｆ	13/16	５１０	Ｇ０６Ｆ 13/16	５１０Ｄ	
	15/16	６１０	15/16	６１０Ａ	